AFLOKKAT

Matière : Symfony

Date : 01/02/2023

Formateur : Utrera Ludovic

Auteur : Utrera Ludovic

SYMFONY : BDD, Admin et Template

SOMMAIRE

AFLOKKAT

AFLOKKAT

01

Mettre en place une base de données

01 Mettre en place une base de données

AFLOKKAT

Le site web du livre d'or de la conférence permet de recueillir des commentaires pendant les conférences. Nous avons besoin de stocker ces commentaires dans un stockage persistant. Un commentaire est mieux décrit par une structure de données fixe : un nom, un email, le texte du commentaire et une photo facultative. Ce type de données se stocke facilement dans un moteur de base de données relationnelle traditionnel.

Nous allons utiliser le moteur de BDD MySQL

01 Mettre en place une base de données

AFLOKKAT

Lancer votre serveur de BDD (Mamp ou Wamp par exemple...).

Ensuite vous devez modifier le fichier .env et notamment la ligne suivante en rajoutant vos paramètres.

```
DATABASE_URL="mysql://user:password@host:port/dbname?serverVersion&charset=utf8mb4
"
```

Dans dbname, renseignez le nom que vous souhaitez pour votre BDD, (ici par exemple sampleconfdb)

Utilisez ensuite la commande suivante depuis votre terminal pour créer votre base de données :

```
php bin/console doctrine:database:create
```

01 Créer des classes d'entités

AFLOKKAT

Une conférence peut être décrite en quelques propriétés :

- **La ville où la conférence est organisée**
- **L'année de la conférence**
- **Une option international pour indiquer si la conférence est locale ou internationale**

Le Maker Bundle peut nous aider à générer une classe (une entité) qui représente une conférence.
Il est maintenant temps de générer l'entité Conférence :

```
symfony console make:entity Conference
```

01 Créer des classes d'entités

AFLOKKAT

Maintenant, nous allons générez une classe d'entité pour les commentaires de la conférence :

symfony console make:entity Comment

01 Lier les entités

AFLOKKAT

Les deux entités, Conference et Comment, devraient être liées l'une à l'autre. Une conférence peut avoir zéro commentaire ou plus et un commentaire est forcément associé à une conférence.

Nous allons utilisez à nouveau la commande make:entity pour ajouter cette relation à la classe Conference en utilisant un type très utile ! relation

```
symfony console make:entity Conference
```

Tout ce dont vous avez besoin pour gérer la relation a été généré pour vous. Une fois généré, le code devient le vôtre ; n'hésitez pas à le personnaliser comme vous le souhaitez

Si vous entrez ? comme réponse pour le type, vous obtiendrez tous les types pris en charge par la commande

01 Lier les entités

AFLOKKAT

Les deux entités, Conference et Comment, devraient être liées l'une à l'autre. Une conférence peut avoir zéro commentaire ou plus et un commentaire est forcément associé à une conférence.

Nous allons utilisez à nouveau la commande make:entity pour ajouter cette relation à la classe Conference en utilisant un type très utile ! relation

```
symfony console make:entity Conference
```

Tout ce dont vous avez besoin pour gérer la relation a été généré pour vous. Une fois généré, le code devient le vôtre ; n'hésitez pas à le personnaliser comme vous le souhaitez

Si vous entrez ? comme réponse pour le type, vous obtiendrez tous les types pris en charge par la commande

01 Ajouter d'autres propriétés (Excercice)

AFLOKKAT

Je viens de réaliser que nous avons oublié d'ajouter une propriété sur l'entité Comment : une photo de la conférence peut être jointe afin d'illustrer un retour d'expérience.
Exécutez à nouveau make:entity et ajoutez une propriété photoFilename de type string. Mais, comme l'ajout d'une photo est facultatif, permettez-lui d'être null :

01 Migrer la base de données

AFLOKKAT

La structure du projet est maintenant entièrement décrite par les deux classes générées.

Ensuite, nous devons créer les tables de base de données liées à ces entités PHP.
Doctrine Migrations est la solution idéale pour cela. Le paquet a déjà été installé dans le cadre de la dépendance orm.

Une migration est une classe qui décrit les changements nécessaires pour mettre à jour un schéma de base de données, de son état actuel vers le nouveau, en fonction des attributs de l'entité. Comme la base de données est vide pour l'instant, la migration devrait consister en la création de deux tables.
Voyons ce que Doctrine génère :

```
symfony console make:migration
```

Notez le nom du fichier généré (un nom qui ressemble à migrations/Version20191019083640.php)

01 Mettre à jour la BDD

AFLOKKAT

Vous pouvez maintenant exécuter la migration générée pour mettre à jour le schéma de la base de données :

```
symfony console doctrine:migrations:migrate
```

Le schéma de la base de données locale est à jour à présent, prêt à stocker des données.

AFLOKKAT

02 Configurer une interface d’administration

02

Configurer une interface d'administration

AFLOKKAT

L'ajout des prochaines conférences à la base de données est le travail des admins du projet. Une interface d'administration est une section protégée du site web où les admins du projet peuvent gérer les données du site web, modérer les commentaires, et plus encore.

Il existe un moyen de créer cette interface très rapidement. Comment ? En utilisant un bundle capable de générer une interface d'administration basée sur la structure du projet. EasyAdmin.

02

Installer des dépendances supplémentaires

AFLOKKAT

Même si le package webapp a ajouté automatiquement de nombreux packages utiles, pour des fonctionnalités plus spécifiques, nous devons ajouter d'autres dépendances ? Avec Composer. En plus des paquets « standards » de Composer, nous travaillerons avec deux types de paquets « spéciaux » :

Composants Symfony : Paquets qui implémentent les fonctionnalités de base et les abstractions de bas niveau dont la plupart des applications ont besoin (routage, console, client HTTP, mailer, cache, etc.) ;

Bundles Symfony : Paquets qui ajoutent des fonctionnalités de haut niveau ou fournissent des intégrations avec des bibliothèques tierces (les bundles sont principalement créés par la communauté).

02 Installer des dépendances supplémentaires

AFLOKKAT

Ajoutez EasyAdmin comme dépendance du projet : symfony composer req "admin:^4«

admin est un alias pour le paquet easycorp/easyadmin-bundle.

Les alias ne sont pas une fonctionnalité interne à Composer, mais un concept fourni par Symfony pour vous faciliter la vie. Les alias sont des raccourcis pour les paquets populaires de Composer. Vous voulez un ORM pour votre application ? Demandez orm. Vous voulez développer une API ? Demandez api. Ces alias font référence à un ou plusieurs paquets normaux de Composer. Ce sont des choix arbitraires faits par l'équipe principale de Symfony.

Un autre détail intéressant est que vous pouvez toujours omettre le symfony du nom des paquets. Demandez cache au lieu de symfony/cache.

Vous souvenez-vous que nous avons mentionné un plugin Composer nommé symfony/flex ? Les alias sont l'une de ses fonctionnalités.

02 Configurer EasyAdmin

AFLOKKAT

EasyAdmin crée automatiquement une section d'administration pour votre application basée sur des contrôleurs spécifiques.

Pour débuter avec EasyAdmin, commençons par générer un "tableau de bord d'administration" qui sera le point d'entrée principal pour gérer les données du site.

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

02 Configurer EasyAdmin

AFLOKKAT

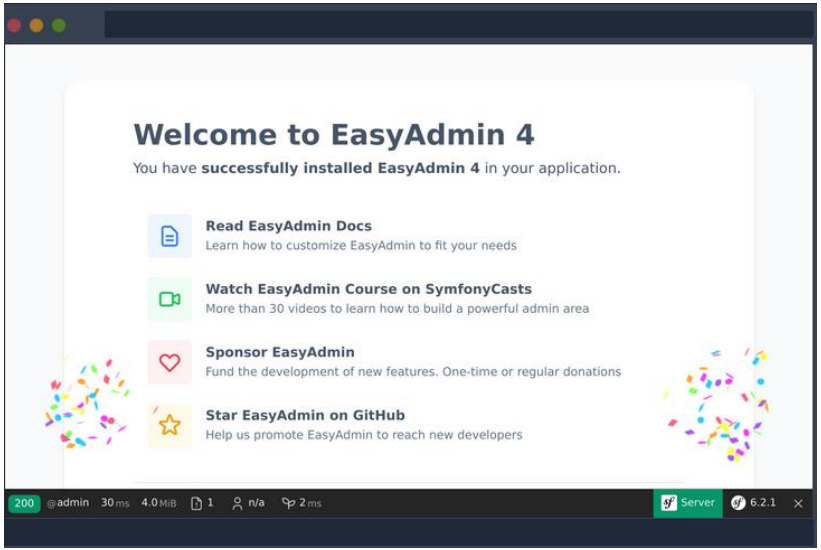
Par convention, les contrôleurs d'administration sont stockés dans leur propre espace de nom App\Controller\Admin.

Accédez à l'interface d'administration générée grâce à l'URL /admin telle que configurée par la méthode index() (vous pouvez modifier l'URL comme bon vous semble) :

02

Configurer EasyAdmin

AFLOKKAT



Et voila ! Nous avons une belle interface d'administration, prête à être adaptée à nos besoins.

L'étape suivante consiste à créer des contrôleurs pour gérer les conférences et les commentaires.

02

Configurer EasyAdmin

AFLOKKAT

Dans le contrôleur du tableau de bord, vous avez peut-être remarqué la méthode configureMenuItems() qui contient un commentaire à propos de l'ajout de liens aux "CRUDs". "CRUD" est un acronyme pour "Create, Read, Update and Delete", les quatre opérations de base que vous allez effectuer sur une entité. C'est exactement ce que nous voulons que notre page d'administration fasse pour nous. EasyAdmin facilite encore plus les choses en prenant en charge les fonctionnalités de filtre et de recherche.

02 Configurer EasyAdmin

AFLOKKAT

Générons un CRUD pour les conférences :

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

Sélectionnez 1 pour créer une interface d'administration pour les conférences et utiliser les valeurs par défaut pour les autres questions. Le fichier src/Controller/Admin/ConferenceCrudController.php devrait être généré :

02 Configurer EasyAdmin

AFLOKKAT

Faites la même chose pour les commentaires :

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

La dernière étape consiste à relier les CRUDs d'administration des conférences et des commentaires au tableau de bord :

02 Configurer EasyAdmin

AFLOKKAT

Pour ça, Nous allons surcharger la méthode `configureMenuItems()` pour ajouter les éléments de menu avec les icônes adéquates pour les conférences et les commentaires, et pour ajouter un lien de retour vers la page d'accueil du site.

EasyAdmin expose une API pour faciliter les liaisons avec les CRUDs des entités via la méthode `MenuItem::linkToRoute()`.

Le tableau de bord principal est vide pour le moment. C'est ici que vous pouvez afficher certaines statistiques, ou n'importe quelle information pertinente. Comme nous n'avons rien d'important à y afficher, redirigeons cette page vers la liste des conférences :

02 Configurer EasyAdmin

AFLOKKAT

Quand nous affichons les relations entre les entités (la conférence liée à un commentaire), EasyAdmin essaie d'utiliser la représentation textuelle de la conférence. Par défaut, il s'appuie sur une convention qui utilise le nom de l'entité et la clé primaire (par exemple Conference #1) si l'entité ne définit pas la méthode "magique" `__toString()`. Pour rendre l'affichage plus parlant, ajoutez cette méthode sur la classe Conference :

02 Configurer EasyAdmin

AFLOKKAT

Vous pouvez maintenant ajouter/modifier/supprimer des conférences directement depuis l'interface d'administration. Testez un peu l'interface pour vous familiarisez avec et ajoutez au moins une conférence.

Ensuite, Ajoutez quelques commentaires sans photos. Réglez la date manuellement pour l'instant. Nous remplirons la colonne createdAt automatiquement dans une étape ultérieure.

02

Personnaliser EasyAdmin

AFLOKKAT

L'interface d'administration par défaut fonctionne bien, mais elle peut être personnalisée de plusieurs façons pour améliorer son utilisation. Faisons quelques changements simples pour montrer quelques possibilités :

Pour personnaliser la section Commentaire, lister les champs de manière explicite dans la méthode `configureFields()` nous permet de les ordonner comme nous le souhaitons. Certains champs bénéficient d'une configuration supplémentaire, comme masquer le champ texte sur la page d'index.

La méthode `configureFilters()` définit quels filtres apparaissent au dessus du champ de recherche.

AFLOKKAT

03 Construire l'interface graphique

03 Construire l'interface graphique

AFLOKKAT

Tout est maintenant en place pour créer la première version de l'interface du site. On ne la fera pas jolie pour le moment, seulement fonctionnelle.

Vous vous souvenez de l'échappement de caractères que nous avons dû faire dans le contrôleur, pour l'easter egg, afin d'éviter les problèmes de sécurité ? Nous n'utiliserons pas PHP pour nos templates pour cette raison. À la place, nous utiliserons Twig. En plus de gérer l'échappement de caractères, Twig apporte de nombreuses fonctionnalités intéressantes, comme l'héritage des modèles.

03 Twig : présentation

AFLOKKAT

Twig est un moteur de modèles moderne pour PHP

Ces principaux avantages sont les suivants :

- **Rapide : Twig compile les modèles en un simple code PHP optimisé. La surcharge par rapport au code PHP normal a été réduite au minimum.**
- **Sécurisé : Twig dispose d'un mode sandbox pour évaluer le code de modèle non fiable. Cela permet à Twig d'être utilisé comme un langage de modèles pour les applications où les utilisateurs peuvent modifier la conception du modèle.**
- **Flexible : Twig est alimenté par un lexer et un analyseur flexible. Cela permet au développeur de définir ses propres balises et filtres personnalisés, et de créer son propre DSL.**

03

Twig VS PHP : Concis

AFLOKKAT

Le langage PHP est verbeux et devient ridiculement verbeux lorsqu'il s'agit de l'échappement des données. En comparaison, Twig a une syntaxe très concise, ce qui rend les modèles plus lisibles :

```
<?php echo $var ?>
<?php echo htmlspecialchars($var,
ENT_QUOTES, 'UTF-8') ?>
```

```
{{ var }}
{{ var|escape }}
{{ var|e }}
```

03 Twig VS PHP : Syntaxe orientée modèle

AFLOKKAT

Twig a des raccourcis pour les modèles communs, comme avoir un texte par défaut affiché lorsque vous itérez sur un tableau vide :

```php
<?php
  if(count($users) > 0){
    foreach($users as $user){
      echo $user->name;
    }
  }
  else{
    echo "No users have been found";
  }
?>
```

```twig
{% for user in users %}
   * {{ user.name }}
{% else %}
   No users have been found.
{% endfor %}
```

03 Twig VS PHP : Full Featured

AFLOKKAT

Twig prend en charge tout ce dont vous avez besoin pour créer facilement des modèles puissants : héritage multiple, blocs, mise en forme automatique de la sortie, et bien plus encore :

```
{% extends "layout.html" %}

{% block content %}
    Content of the page...
{% endblock %}
```

03 Utiliser Twig

AFLOKKAT

Toutes les pages du site Web suivront le même modèle de mise en page, la même structure HTML de base. Lors de l'installation de Twig, un répertoire templates/ a été créé automatiquement, ainsi qu'un exemple de structure de base dans base.html.twig.

Un modèle peut définir des blocks. Un block est un emplacement où les templates enfants, qui étendent le modèle, ajoutent leur contenu.

Créons un template pour la page d'accueil du projet dans templates/conference/index.html.twig :

03 Utiliser Twig

AFLOKKAT

```
{% extends 'base.html.twig' %}

{% block title %}Conference Guestbook{% endblock %}

{% block body %}
    <h2>Give your feedback!</h2>

    {% for conference in conferences %}
        <h4>{{ conference }}</h4>
    {% endfor %}
{% endblock %}
```

Le template étend (ou extends) base.html.twig et redéfinit les blocs title et body.

La notation {% %} dans un template indique des actions et des éléments de structure.

La notation {{ }} est utilisée pour afficher quelque chose. {{ conference }} affiche la représentation de la conférence (le résultat de l'appel à la méthode__toString de l'objet Conference).

03 Utiliser Twig dans un Controller

AFLOKKAT

Nous allons mettre à jour le code de notre Controller pour générer le bon contenu dans notre template Twig.

Nous avons besoin dans notre controller d'appeler le template Twig en passant un paramètre, la syntaxe est la suivante :

```
return new Response($twig->render('conference/index.html.twig', [
        'conferences' => $conferenceRepository->findAll(),
 ]));
```

Pour pouvoir générer le contenu du template, nous avons besoin de l'objet Environment de Twig (le point d'entrée principal de Twig). Notez que nous demandons l'instance Twig en spécifiant son type dans la méthode du Controller. Symfony est assez intelligent pour savoir comment injecter le bon objet.

Nous avons également besoin du repository des conférences pour récupérer toutes les conférences depuis la base de données.

Dans le code du contrôleur, la méthode render() génère le rendu du template et lui passe un tableau de variables. Nous passons la liste des objets Conference dans une variable conferences.

03 Créer la page d'une conférence

AFLOKKAT

Chaque conférence devrait avoir une page dédiée à l'affichage de ses commentaires. L'ajout d'une nouvelle page consiste à ajouter un contrôleur, à définir une route et à créer le template correspondant.

Nous allons ajouter une méthode show() dans le fichier src/Controller/ConferenceController.php :

La dernière étape consiste à créer le fichier templates/conference/show.html.twig :

Dans ce template, nous utilisons le symbole | pour appeler les filtres Twig. Un filtre transforme une valeur. comments|length retourne le nombre de commentaires et comment.createdAt|format_datetime('medium', 'short') affiche la date dans un format lisible par l'internaute.

Pour utiliser la méthode format_datetime nous avons besoin d'installer la dépendance suivante :

```
symfony composer req "twig/intl-extra:^3"
```

03 Lier des pages entre elles

AFLOKKAT

La toute dernière étape pour terminer notre première version de l'interface est de rendre les pages de la conférence accessibles depuis la page d'accueil :

Nous pourrions coder le chemin en dur dans le template, mais est-ce vraiment une bonne idée ?

03 Lier des pages entre elles

AFLOKKAT

Twig fourni la méthode path() qui prend en paramètre le nom d'une route. Un des avantages d'utiliser cette fonction c'est que si demain vous modifiez l'URL de la route vous n'êtes pas obligé de modifier tous les chemins.

03 Paginer les commentaires

AFLOKKAT

Imaginons que pour une conférence donnée, nous ayons un nombre important de commentaires. Il s'agît d'un cas très fréquent en informatique. Symfony met un outil à notre disposition pour gérer ce scénario. Le paginator.

03 Paginer les commentaires

AFLOKKAT

Créez une méthode `getCommentPaginator()` dans `CommentRepository`. Cette méthode renvoie un `Paginator` de commentaires basé sur une conférence et un décalage (où commencer) :

Nous allons fixé le nombre maximum de commentaires par page à 2 pour faciliter les tests.

Pour gérer la pagination dans le template, transmettez à Twig le Doctrine Paginator au lieu de la Doctrine Collection :

03 Optimiser le controller

AFLOKKAT

Vous avez peut-être remarqué que les deux méthodes présentes dans ConferenceController prennent un environnement Twig comme argument. Au lieu de l'injecter dans chaque méthode, appelons la méthode render() de la classe parente :

04 Conclusion

AFLOKKAT

Dans cette partie nous avons vu :

Comment mettre en place une base de données
Comment construire une interface d'administration
Comment mettre en place une interface graphique avec Twig

Le seul problème de notre interface d'administration pour le moment c'est que n'importe qui peut y accéder. Nous allons voir comment nous pouvons sécuriser l'interface dans la prochaine partie.

AFLOKKAT

Sources

https://symfony.com
https://stackoverflow.com/
https://openclassrooms.com
https://twig.symfony.com/

MERCI DE VOTRE ATTENTION !

Quelques icônes pour aider

AFLOKKAT