Movies! | Symfony3

Présentation

Movies! est un site de recommandation de films. Les utilisateurs n'y trouvent que des films de qualité, consultent des informations sur ceux-ci (dont les bandes annonces), laissent des critiques publiques, et peuvent ajouter des films à leur "liste de films à voir".

Un back-office permet aux administrateurs de gérer la base de données de films, de critiques et d'utilisateurs du site.

Matériel fourni

La base de données de films est fourni dans un fichier .sql.
Les entités correspondantes ont déjà été générées et sont également fournie, ainsi que les affiches (posters) des films.

Technos utilisées

- Serveur : Apache
- Base de données : MySQL
- Backend : PHP7 / Symfony3
- ORM : Doctrine
- Moteur de template : Twig
- Framework CSS : aucun
- Gestion de versions : git

Conseils pour la mise en place du projet :

1. Installer un nouveau projet Symfony (à partir du fichier sf3.zip du réseau)
2. Créer votre base de donnée à partir de PHPMyAdmin en important le fichier .sql
3. Configurer la connexion à la base de donnée dans app/config/parameters.yml
4. Effacer le bundle puis générer un nouveau AppBundle
5. Créer un MovieController en dupliquant le DefaultController (renommer le fichier et la classe).
6. Créer un base.html.twig à vos envies. Y inclure au moins une feuille de style.
7. Placer les “posters” dans votre dossier web/.
8. Placer les entités et les repos fournis dans votre bundle
9. Faire un premier commit (git)

Étapes à suivre

1. ACCUEIL

La page d'accueil présente une liste d'affiches de film. Il n'y a que les affiches qui sont affichées, pour un look bien graphique.
Chacune est cliquable, et mène vers la page de détails du film.

Par défaut, 50 affiches sont affichées (les 50 films les plus récents). Un système de pagination sera mis en place plus tard.

Conseils

- Cette page est à créer dans le MovieController.
- Pour récupérer 50 films, utiliser la méthode findBy() du repository. Le 3e argument est le nombre de lignes à récupérer. Le premier peut être un tableau vide pour l'instant, et le second est l'ordre sous forme de tableau.
- N'oubliez pas de passer les films à Twig pour pouvoir les afficher (dans le 2e argument de la méthode render()) !
- Avant de pouvoir faire des liens sur chaque affiche, vous devez avoir déjà créé la route de la page détail.

2. PAGE DE DÉTAILS

La page de détail d'un film affiche le poster, le titre, l'année, et toutes les autres informations publiques que nous possédons sur le film.

La bande-annonce (trailer) est à intégrer directement dans la page.

Conseils

- Cette page est à créer dans le MovieController
- Vous avez besoin de l'id du film dans la route ! Utiliser une URL de type details/{id}
- Récupérer le film avec la méthode find($id) du repository
- Vous avez l'identifiant YouTube du trailer de chaque film. Pour savoir comment intégrer la bande-annonce sur votre site, analyser les codes fournis par YouTube !

3. BACK-OFFICE

Créer un back-office permettant de gérer les films. Celui-ci devrait vaguement ressembler à la page d'affichage d'une table de PHPMyAdmin. En effet, une liste sous forme de tableur devrait présenter les films. Sur chaque ligne, un bouton doit permettre de supprimer le film, et un autre de le modifier.

Un lien au-dessus de ce tableau devrait permettre d'ajouter un nouveau film.

Note : il n'est pas nécessaire de pouvoir gérer les catégories, elles sont fixes.

Conseils

- Créer un nouveau contrôleur : AdminMovieController
- Placer toutes les urls de ce contrôleur sous le préfixe /admin/ afin de faciliter le contrôle d'accès plus tard.
- Pour l'ajout ou la modification des films, générer un formulaire avec doctrine:generate:form. Utiliser la même classe de formulaire pour les 2 actions.
- La page d'ajout d'un film présente un formulaire vide.
- La page de modification d'un film présente un formulaire prérempli. Cette page possède l'id du film dans l'url. Dans le contrôleur, le film est d'abord récupéré en bdd, puis est associé au formulaire.
- Créer une route spécifique pour supprimer un film. Utiliser la méthode ->remove() du Manager dans le contrôleur. Il est nécessaire de d'abord récupérer le film avant de pouvoir le supprimer !

4. CRITIQUES DE FILMS

Sur la page de détail des films, afficher un formulaire. Celui-ci permet aux utilisateurs du site de publier des critiques sur le film.
Les critiques sont toutes affichées sous ce formulaire.

Le formulaire doit permettre de renseigner :

- Un titre pour la critique
- La critique elle-même
- Une note sur 10
- Un pseudo

Tous les champs sont requis.

Conseils

- Créer une nouvelle entité : Review
- Associer cette entité aux Movie (relation bi-directionnelle)
- Traiter le formulaire directement dans votre de méthode de contrôleur qui affiche la page de détails
- Il n’est pas nécessaire de faire une requête pour récupérer les critiques du film !

5. BACK-OFFICE, SUITE

Exactement dans le même esprit que pour la gestion des films, l'administrateur doit pouvoir gérer les Reviews (voir, créer, modifier, supprimer).

Conseils

- Préfixer toujours vos routes par /admin/
- Créer 1 contrôleur : AdminReviewController
- La suppression d'un Movie doit également supprimer les Reviews associées !

6. PAGINATION EN PAGE D'ACCUEIL

Ajouter un système de pagination à la page d'accueil. Il doit ainsi être possible de :

- naviguer vers la "page suivante" ou la "page précédente" grâce à 2 liens. Ces liens ne doivent être présents (ou actifs) que s'il y a lieu.
- afficher le numéro des films actuellement affichés, sur le total de films (du genre : "affichage des films #51 à 100 sur 357 films").

Étapes de réalisation :

1. Modifier votre route de la page d'accueil pour qu'elle contienne un paramètre d'URL {page}. Donner une valeur par défaut de 1 à ce paramètre. S'inspirer fortement de : http://symfony.com/doc/current/routing.html#giving-placeholders-a-default-value
2. Utiliser le 4e argument de la méthode findBy() pour modifier l'offset (le numéro du premier film récupéré). La valeur à mettre ici est le numéro de page - 1 * nombre de films par page !
3. Pour afficher les liens *Page suivante* et *Page précédente,* vous devez :
    a. Dans le contrôleur, créer une variable contenant le numéro de la page suivante (page + 1)
    b. Créer une variable pour la page précédente (page - 1)
    c. Passer ces 2 variables à Twig
    d. Utiliser ces variables pour générer les liens
4. Pour compter le nombre total de films présents dans la base, vous devez :
    a. Créer une nouvelle méthode dans votre MovieRepository (ie. countAll())
    b. Ajouter ce code dans la méthode : http://stackoverflow.com/a/9215880. Changer *account* pour *movie...*
    c. Retourner le résultat dans la méthode
    d. Appeler cette méthode depuis le MovieController et récupérer la valeur dans une variable
    e. Passer cette variable à Twig

7. UTILISATEURS

Permettre aux utilisateurs du site de :

- se créer un compte
- se connecter
- se déconnecter

Une fois connecté, un message doit indiquer "Bonjour pseudo !" dans l'entête du site, ainsi qu'un lien de déconnexion.
Sinon, un lien menant vers l'inscription, et un autre vers la connexion doivent s'afficher.

Lorsque déconnecté, un lien vers la page de connexion, et un autre vers la page d'inscription sont affichés dans l'entête.

Les pages d'accueil et de détails sont accessibles à tous les utilisateurs (connectés ou non).

Il y aura 2 types d'utilisateurs : les simples users du site, et les administrateurs.

Conseils

- Contrairement à Nantes Nocturne, implémenter correctement les rôles. Celui-ci doit être sauvegardé en bdd. Au moment de l'inscription, affecter toujours un rôle ROLE_USER.
- Pour le reste, réutiliser tout le code des utilisateurs de Nantes Nocturne. Pour mémoire, vous avez a minima besoin de :
    - app/config/security.yml
    - AppBundle/Controller/UserController.php
    - L'entité User.php, qui doit absolument implémenter UserInterface
    - Le formulaire d'inscription
    - Les vues twig pour le login, l'inscription, et les mots de passe oublié

8. LISTE DES FILMS À VOIR

Sur la page de détail d'un film, un bouton doit permettre d'ajouter un film à sa WatchList, sa liste personnelle de films à voir. Ce bouton ne s'affiche que pour les utilisateurs connectés. Si le film est déjà présent dans sa WatchList, le bouton permet alors de retirer le film.

Un lien dans l'entête permet de se rendre sur sa WatchList. Celle-ci affiche la liste de ses films à voir, dans la même présentation qu'en page d'accueil.

Conseils

- Créer une relation ManyToMany entre les Movie et les User !

9. BACK-OFFICE, FIN

Exactement dans le même esprit que pour la gestion des films, l'administrateur doit pouvoir gérer les Users (voir, créer, modifier, supprimer).

Le formulaire de création/modification pour les Users doit permettre de changer le rôle d'un utilisateur (ROLE_USER ou ROLE_ADMIN).

Sécuriser le back-office : seuls les administrateurs doivent pouvoir y accéder.

Conseils

- Préfixer toujours vos routes par /admin/
- Créer 1 contrôleur : AdminUserController
- La suppression d'un User doit également supprimer sa WatchList !

10. RECHERCHE ET FILTRES

Sur la page d'accueil, ajouter un formulaire permettant de rechercher un film par mots-clefs. La recherche doit se faire dans les champs que vous jugerez pertinent, mais au minimum dans "title", "directors" et "actors".

Dans le même formulaire, ajouter une liste déroulante de toutes les catégories des films, qui permet de n'afficher que les films de la catégorie choisie.

Sur soumission du formulaire, la pagination doit revenir à 1. Les données soumises doivent être présentes dans l'URL sous forme de paramètres d'URL normaux (ie. ?genre=12&keyword=brad). Le formulaire doit donc être en GET.

Conseils

- La méthode findBy() ne suffit plus maintenant… vous devez utiliser le QueryBuilder ou du DQL. Voir par exemple : http://symfony.com/doc/2.8/doctrine.html#querying-for-objects-with-dql
- Le composant Form de Symfony peut s'avérer davantage une nuisance qu'une aide sur ce genre de tâches… utilisez les bons vieux formulaires HTML :)