Kata Symfony

Aurélien Feuillard

Le kata Symfony consiste à créer une application de gestion de films.

**Toutes les étapes** doivent faire l’objet d’**un commit et d’un push** sur la branche qui t’a été confiée.

La documentation est essentiellement celle de Symfony.com

Pour faciliter le démarrage, des informations précises et les documentations concernées sont indiquées, ce n’est plus le cas dans la suite où l’objectif est d’amener le développeur à trouver ses propres ressources.

### Étape 1 : Initialisation du projet Symfony

Crée un nouveau projet Symfony avec la commande composer et le package symfony en version 7.1

Installe les dépendances requises (Doctrine, Twig, Form).

Instancie le dépôt git local

Crée une branche develop

Commit et push le projet instancié sur le repository qui t’a été partagé

[*Documentation officielle*](https://symfony.com/doc/current/setup.html)

### Étape 2 : Configuration de la base de données

Configure une base de données SQLite pour Doctrine.

Lance la création de la base de données avec la commande symfony appropriée.

Le fichier de la base de données doit se trouver dans ./var/storage/db.sqlite

[*Documentation officielle*](https://symfony.com/doc/current/doctrine.html)

### Étape 3 : Création de l’entité Film

Crée une entité *movie* avec les champs title, year, type, abstract.

*type* sert à indiquer le genre du film

*abstract* permet d’indiquer le résumé du film

Tu peux utiliser la commande maker ou créer ton fichier from scratch en suivant les exemples de la documentation

[*Documentation officielle doctrine pour Symfony*](https://symfony.com/doc/current/doctrine.html)

[Documentation des attributs dans Doctrine](https://www.doctrine-project.org/projects/doctrine-orm/en/2.16/reference/attributes-reference.html#attrref_column)

Lance les migrations pour créer la table correspondante.

Vérifie que la table *movie* a été correctement créée et indique la méthode que tu as retenu pour visualiser le contenu de la bdd

### Étape 4 : Page d’accueil

Crée une route qui s’appelle *homepage* pour une page d’accueil (/), où tu affiches un titre h1 Movie List et un texte *Coming soon, my favorite movies*

Tu peux utiliser la commande maker pour créer le controller.

Vérifie le code généré et adapte le template twig

[Documentation officielle](https://symfony.com/doc/current/controller.html)

### Étape 5 : Frontend avec Bootstrap

Utilise Webpack encore et Bootstrap pour embellir le frontend.

### Étape 6 : Formulaire d’ajout de film

Crée un formulaire pour ajouter un film manuellement.

Affiche le formulaire sur une nouvelle route /film/add.

Valide et traite les données soumises.

### Étape 7 : Affichage des films

Sur la page d'accueil, affiche la liste des films ajoutés dans la base de données.

Utilise Twig pour présenter les données.

### Étape 8 : Détail d’un film

Crée une route /film/{id} qui affiche les détails d’un film.

Affiche les informations stockées dans la base de données sur cette page.

### Étape 9 : Édition d’un film

Crée un formulaire pour modifier un film existant.

Affiche le formulaire sur une route /film/edit/{id}.

Gère la validation et la mise à jour des informations.

### Étape 10 : Suppression d’un film

Implémente la suppression d’un film via un bouton sur la page de détail.

Ajoute une confirmation avant la suppression.

### Étape 11 : Rechercher un film via l’api OMDB

Crée une nouvelle route /omdb sur laquelle l’utilisateur peut rechercher des films dans la base OMBD.

Développe un client OMDB avec le composant httpClient

Récupère une clé api gratuite sur <https://www.omdbapi.com>

Indique la clé api en variable d’environnement dans .env.local afin que le client puisse appeler OMDB

Affiche le nombre de films trouvé (total) et affiche les informations de chaque film retourné par l’api

Pour cela, utilise une classe DTO pour stocker le retour de l’API Json vers un objet *MovieDTO*

### Étape 12 : Ajout simplifié

Ajoute une fonctionnalité “Ajouter à mes films” pour enregistrer un film dans ta bibliothèque.

### Étape 13 : Ajout du poster

Modifie l’application pour stocker l’url du poster d’un film et affiche le poster dans l’application ou une image par défaut.

### Étape 14 : Utilisation d’un workflow

Ajoute un champ *lastPlayedAt* qui permet d’indiquer la dernière date de visionnage. Le champ est nullable.

Ajoute un champ *status* qui permet de stocker le statut d’un film avec les seules valeurs *unseen* et *seen*, la valeur par défaut est *unseen*.

Installe le composant Symfony Workflow.

Configure un workflow simple pour le changement d’état unseen vers seen

Mets en place un bouton pour changer le statut d’un film sur la page de détail.

Lorsqu’un film change d’état et devient *seen*, utilise une transition pour mettre à jour le champ *lastPlayedAt* avec le datetime présent. Si le status repasse à *unseen* le champs *lastPlayedAt* reprend la valeur *null*

### Étape 15 : Tester le client API OMDb

Crée des tests unitaires pour le service qui interagit avec l’API OMDb.

Assure-toi que toutes les méthodes clés sont testées.