Documentation technique Symfony

Guilbaud Clément
Delhaye Gauvain

Sommaire

Connexion à la base de données.

Host: bcr5zyonhb2yqvejnpon-mysql.services.clever-cloud.com

Database name: bcr5zyonhb2yqvejnpon

User: ulziuasxpugdrrzg

Password: nv8n1gwGPTAWC8cDEQmX

Port: 3306

Connection uri:
mysql://ulziuasxpugdrrzg:nv8n1gwGPTAWC8cDEQmX@bcr5zyonhb2yqvejnpon-mysql.services.clever-cloud.com:3306/bcr5zyonhb2yqvejnpon

Lancement de la base de données.

Dans un premier temps il faut lancer le serveur symfony: *symfony server:start*

Ensuite nous utilisons la commande: *php bin/console doctrine:fixture:load* qui permet de peupler la base de données avec des données prédéfinies, ce qui est très utile pour le développement ou les tests automatisés. Elle nous assure que nous avons un ensemble de données connu et constant avec lequel travailler

Méthodes dans RegistrationController

1. construct(RegistrationService $registrationService)
    - **Paramètres :** RegistrationService $registrationService
    - **Description :** Ce constructeur initialise le contrôleur avec les services nécessaires pour gérer les inscriptions. Les commentaires spécifiques ne sont pas fournis dans le code.
2. show($id)
    - **Paramètres :** Extrapolation difficile due à une extraction imparfaite, mais il semble traiter de l'affichage des inscriptions par ID de tournoi.
    - **Description :** Récupère et retourne les inscriptions associées à un tournoi spécifié par ID. Réponse sous forme de JSON. La méthode semble liée à une route API pour obtenir les détails d'inscription.
3. register(Request $request, int $tournamentId)
    - **Paramètres :** Request $request, int $tournamentId
    - **Description :** Permet l'inscription d'un utilisateur à un tournoi. Les détails de l'utilisateur et du tournoi sont obtenus via la requête, et une réponse JSON est générée. Cela correspond à une route POST pour créer une nouvelle inscription.

4. cancel(int $idTournament, int $idRegistration)
    - **Paramètres :** int $idTournament, int $idRegistration
    - **Description :** Annule une inscription donnée, vraisemblablement vérifie aussi l'utilisateur qui fait la demande. Répond avec un message JSON confirmant l'annulation.

Méthodes dans TournamentController

1. construct(TournamentRepository $tournamentRepository)
    - **Paramètres :** TournamentRepository $tournamentRepository
    - **Description :** Ce constructeur initialise le contrôleur avec les services de répertoire nécessaires pour gérer les tournois.
2. list()
    - **Paramètres :** Aucun
    - **Description :** Récupère et retourne tous les tournois enregistrés dans la base de données sous forme de réponse JSON.
3. show($id)
    - **Paramètres :** $id (l'identifiant du tournoi)
    - **Description :** Retourne les détails d'un tournoi spécifique par son ID. Gère également les cas où le tournoi n'existe pas, renvoyant une réponse 404.
4. create(Request $request, TournamentService $tournamentService)
    - **Paramètres :** Request $request, TournamentService $tournamentService
    - **Description :** Permet de créer un nouveau tournoi avec les données fournies dans la requête. Valide les données et renvoie une réponse JSON appropriée.
5. update($id, Request $request)
    - **Paramètres :** $id, Request $request
    - **Description :** Met à jour les détails d'un tournoi existant. Gère les cas où le tournoi spécifié n'existe pas.
6. delete($id)
    - **Paramètres : $id**
    - **Description :** Supprime un tournoi existant par son ID. Comme pour les autres méthodes, il vérifie si le tournoi existe avant de procéder à la suppression.

Méthodes dans UserController

1. construct(UserRepository $userRepository)
    - **Paramètres :** UserRepository $userRepository
    - **Description :** Ce constructeur initialise le contrôleur avec le dépôt nécessaire pour gérer les données des utilisateurs.
2. list()
    - **Paramètres :** Aucun
    - **Description :** Récupère et retourne tous les utilisateurs enregistrés dans la base de données sous forme de réponse JSON.
3. show($id)
    - **Paramètres :** $id (l'identifiant de l'utilisateur)
    - **Description :** Retourne les détails d'un utilisateur spécifique par son ID. Gère également les cas où l'utilisateur n'existe pas, renvoyant une réponse 404.
4. create(Request $request, UserService $userService)
    - **Paramètres :** Request $request, UserService $userService
    - **Description :** Permet de créer un nouvel utilisateur avec les données fournies dans la requête. Valide les données et renvoie une réponse JSON appropriée.
5. update($id, Request $request, UserService $userService)
    - **Paramètres :** $id, Request $request, UserService $userService
    - **Description :** Met à jour les détails d'un utilisateur existant. Gère les cas où l'utilisateur spécifié n'existe pas.
6. delete($id)
    - **Paramètres :** $id
    - **Description :** Supprime un utilisateur existant par son ID. Comme pour les autres méthodes, il vérifie si l'utilisateur existe avant de procéder à la suppression.

Nous pourrons retrouver dans notre base de données différentes entités réparties dans différentes parties.

Entités

• Tournament (Tournoi) :

o tournamentName (string) : Nom du tournoi

o startDate (date) : Date de début du tournoi

o endDate (date) : Date de fin du tournoi

o location (string, optionnel) : Lieu du tournoi

o description (texte) : Description du tournoi

o maxParticipants (integer) : Nombre maximum de participants au tournoi

o status (dynamique) : Le statut du tournoi doit être renvoyé dynamiquement en se basant sur les dates de début et de fin du tournoi.

o sport (string) : Nom du sport

o organizer (relation vers l'entité User) : Organisateur du tournoi

o winner (relation vers l'entité User) : Gagnant du tournoi

o Attention la relation avec Game est au pluriel (games)

• User (Joueur) :

o lastName (string) : Nom de famille du joueur

o firstName (string) : Prénom du joueur

o username (string) : Nom d'utilisateur du joueur

o emailAddress (string) : Adresse email du joueur

o password (string) : Mot de passe du joueur (haché)

o status (string) : Statut de l'utilisateur (actif, suspendu, banni)

• Registration (Inscription) :

o player (relation vers l'entité User) : Joueur inscrit au tournoi

o tournament (relation vers l'entité Tournament) : Tournoi auquel le joueur est inscrit

o registrationDate (date) : Date d'inscription du joueur au tournoi

o status (string) : Statut de l'inscription (confirmée, en attente)

• SportMatch (Partie) :

o tournament (relation vers l'entité Tournament) : Tournoi auquel le match appartient

o player1 (relation vers l'entité User) : Joueur 1 du match

o player2 (relation vers l'entité User) : Joueur 2 du match

o matchDate (date) : Date du match

o scorePlayer1 (integer) : Score du joueur 1 dans le match

o scorePlayer2 (integer) : Score du joueur 2 dans le match

o status (string) : Statut du match (en attente, en cours, terminé)