Introduction

L'authentification est un élément clé de la sécurité d'un projet web. Elle permet de s'assurer que seuls les utilisateurs autorisés peuvent accéder aux différentes parties de l'application. Dans le cadre de notre projet Symfony 6, nous avons choisi d'utiliser le gestionnaire d'utilisateurs intégré à Symfony pour implémenter l'authentification.

Fichiers modifiés

Pour implémenter l'authentification dans notre projet, nous avons dû modifier les fichiers suivants :

- Contrôleur d'authentification : nous avons créé une classe de contrôleur qui gère la connexion et la déconnexion de nos utilisateurs. Cette classe utilise les méthodes de l'interface UserProviderInterface de Symfony pour vérifier les informations de connexion de l'utilisateur et retourner l'objet User correspondant.
- Fichier de configuration de sécurité (security.yaml) : nous avons configuré le système de sécurité de notre projet pour utiliser notre contrôleur d'authentification et notre fournisseur d'utilisateurs. Nous avons également défini des règles de sécurité qui indiquent quelles routes de notre application nécessitent une authentification et comment celle-ci doit être vérifiée.
- Modèle de données : nous avons créé une entité Doctrine pour stocker les informations de nos utilisateurs (nom d'utilisateur, mot de passe, etc.).

Comment s'opère l'authentification

Voici comment s'opère l'authentification dans notre projet Symfony 6 :

1. Lorsqu'un utilisateur souhaite se connecter à notre application, il est redirigé vers notre contrôleur d'authentification.
2. Le contrôleur vérifie les informations de connexion de l'utilisateur (nom d'utilisateur et mot de passe) en utilisant les méthodes de l'interface UserProviderInterface de Symfony.
3. Si les informations de connexion sont correctes, le contrôleur retourne l'objet User correspondant et l'utilisateur est connecté.
4. Si les informations de connexion sont incorrectes, le contrôleur renvoie une erreur et l'utilisateur est invité à réessayer la connexion.

Stockage des utilisateurs

Les utilisateurs sont stockés dans la base de données de ce projet, dans une entité Doctrine dédiée. Vous pouvez utiliser le fournisseur d'utilisateurs EntityUserProvider de Symfony pour accéder à ces données depuis votre contrôleur d'authentification.

Conclusion

Dans ce projet, l'authentification est gérée en utilisant les fonctionnalités de sécurité intégrées de Symfony. Les utilisateurs sont stockés dans la base de données de votre projet, dans une entité Doctrine dédiée, et peuvent être vérifiés en utilisant un fournisseur d'utilisateurs de Symfony. Les routes de l'application qui nécessitent une authentification sont protégées grâce aux annotations de sécurité de Symfony.

Introduction

Authentication is a key element of the security of a web project. It ensures that only authorized users can access different parts of the application. In the context of our Symfony 6 project, we have chosen to use the built-in user manager in Symfony to implement authentication.

Modified files

To implement authentication in our project, we had to modify the following files:

- Authentication controller: we created a controller class that handles the login and logout of our users. This class uses the methods of Symfony's UserProviderInterface to verify the user's login information and return the corresponding User object.
- Security configuration file (security.yaml): we configured the security system of our project to use our authentication controller and our user provider. We also defined security rules that indicate which routes of our application require authentication and how it should be verified.
- Data model: we created a Doctrine entity to store the information of our users (username, password, etc.).

How authentication operates

Here is how authentication works in our Symfony 6 project:

1. When a user wants to login to our application, they are redirected to our authentication controller.
2. The controller verifies the user's login information (username and password) using the methods of Symfony's UserProviderInterface.
3. If the login information is correct, the controller returns the corresponding User object and the user is logged in.
4. If the login information is incorrect, the controller returns an error and the user is prompted to try login again.

User storage

Users are stored in the database of this project, in a dedicated Doctrine entity. You can use Symfony's EntityUserProvider user provider to access this data from your authentication controller.

Conclusion

In this project, authentication is managed using Symfony's built-in security features. Users are stored in the database of your project, in a dedicated Doctrine entity, and can be verified using a Symfony user provider. Routes of the application that require authentication are protected using Symfony's security annotations.