Authentification et autorisations

https://github.com/heig-vd-devprodmed-course/heig-vd-devprodmed-course

Visualiser le contenu complet sur GitHub à cette adresse.

L. Delafontaine, avec l'aide de GitHub Copilot.

Plus de détails sur GitHub

Cette présentation est un résumé du contenu complet disponible sur GitHub.

Pour plus de détails, consulter le contenu complet sur GitHub ou en cliquant sur l'en-tête de ce document.

Objectifs (1)

- Décrire les concepts d'authentification et d'autorisation.
- Stocker et vérifier les mots de passe de manière sécurisée.
- Définir et utiliser la classe `Auth` de Laravel pour gérer l'authentification des utilisateur.trices.

Objectifs (2)

- Définir et utiliser la classe `Hash` de Laravel pour hacher et vérifier les mots de passe.
- Définir et utiliser des gates et des policies pour gérer les autorisations.
- Protéger des routes avec des middlewares d'authentification.
- Associer les ressources aux personnes authentifiées.

Avertissement

Il existe de multiples solutions pour implémenter l'authentification et l'autorisation dans une application Laravel (Fortify, Passport, Sanctum, Socialite).

Dans ce contenu, nous allons utiliser les briques de base fournies par Laravel pour implémenter l'authentification et l'autorisation, afin de permettre de comprendre les concepts fondamentaux et de maîtriser les outils de base avant d'utiliser des solutions plus complexes.

Authentification et autorisations, un rappel

La sécurité d'une application repose sur deux aspects complémentaires :

1. L'authentification.
2. Les autorisations.

Authentification

L'**authentification** est le processus qui permet de vérifier l'identité d'une personne qui accède à un système. C'est la réponse à la question *"Qui êtes-vous ?"*.

Le processus typique :

1. La personne saisit ses informations d'identification.
2. L'application vérifie que les informations sont correctes.
3. Si oui, l'application crée une session.
4. La personne peut accéder aux ressources protégées.

Autorisation

L'**autorisation** est le processus qui permet de déterminer si une personne authentifiée a le droit d'accéder à une ressource ou d'effectuer une action spécifique. C'est la réponse à la question *"Qu'avez-vous le droit de faire ?"*.

L'autorisation intervient après l'authentification : une fois que le système sait qui vous êtes, il détermine ce que vous pouvez faire en fonction de vos permissions, rôles ou règles métier.

Stocker les mots de passe de manière sécurisée

Stocker les mots de passe en clair est dangereux et inapproprié. Il est essentiel de les **hacher** avant de les stocker.

Le hachage est une fonction à sens unique qui transforme une chaîne de caractères en une chaîne de longueur fixe de manière irréversible.

Pour savoir si le mot de passe fourni est correct, on le hash et on compare le hash avec celui stocké en base de données.

Liens avec les sessions

Une **session** est un mécanisme qui permet de maintenir l'état d'une personne entre différentes requêtes HTTP.

Les sessions stockant un identifiant de session (session ID) dans un cookie côté client.

À chaque requête, le serveur utilise cet identifiant pour récupérer les données de session et identifier la personne connectée.

L'application peut ainsi savoir qui est connecté et appliquer les règles d'autorisation en conséquence.

Les classes Auth et Hash

Laravel nous fournit plusieurs classes pour simplifier la gestion de l'authentification la gestion des mots de passe :

- La classe `Auth`.
- La classe `Hash`.

Auth

La façade `Auth` est l'interface principale pour gérer l'authentification dans Laravel.

Méthodes principales :

- `Auth::check()` : vérifier si une personne est authentifiée.
- `Auth::user()` : obtenir la personne authentifiée.
- `Auth::login($user)` : connecter une personne.
- `Auth::attempt($credentials)` : tenter une connexion.
- `Auth::logout()` : déconnecter une personne.

Hash

La classe `Hash` fournit des méthodes pour hacher et vérifier les mots de passe de manière sécurisée.

Méthodes principales :

- `Hash::make('password')` : hacher un mot de passe.
- `Hash::check('password', $hash)` : vérifier un mot de passe.

Plusieurs algorithmes de hachage sont disponibles. Laravel utilise bcrypt par défaut.

Gates, policies et middlewares d'autorisation

Afin de vérifier les autorisations d'une personne authentifiée, Laravel fournit plusieurs outils :

- Les **gates**.
- Les **policies**.
- Les **middlewares**.

Gates (1)

Les **gates** sont des fonctions anonymes qui déterminent si une personne est autorisée à effectuer une action particulière.

Exemple :

```php
Gate::define('admin', function ($user) {
    return $user->is_admin === true;
});
```

Il s'agit du mécanisme de base pour gérer les autorisations dans Laravel, mais très simple et pas très flexible.

Gates (2)

Les gates peuvent ensuite être utilisées pour vérifier les autorisations dans les contrôleurs ou les vues.

```
// Dans un contrôleur
if (Gate::allows('admin')) {
    // La personne est un.e administrateur.trice
}
```

```
// Dans une vue Blade
@can('admin')
    <!-- Afficher du contenu réservé aux administrateurs -->
@endcan
```

Policies (1)

Les **policies** sont des classes qui organisent la logique d'autorisation autour d'un modèle particulier.

Création :

```
php artisan make:policy PostPolicy --model=Post
```

Une nouvelle police sera créée dans le dossier `app/Policies` avec des méthodes pour chaque action (view, create, update, delete, etc.).

Policies (2)

```php
// Méthode dans la classe PostPolicy
public function update(User $user, Post $post): bool
{
    return $user->id === $post->user_id;
}
```

```php
// Utilisation dans le contrôleur PostController
public function update(Request $request, Post $post)
{
    Gate::authorize('update', $post);

    // ...
}
```

Middlewares (1)

Les **middlewares** sont des filtres qui s'exécutent avant ou après une requête HTTP.

Ils peuvent vérifier ou modifier la requête ou la réponse HTTP et sont souvent utilisés pour protéger des routes.

Le middleware `auth` vérifie que la personne est authentifiée avant d'autoriser l'accès à une route.

Si la personne n'est pas connectée, elle sera redirigée vers la page de connexion.

Middlewares (2)

```php
// Protéger une route particulière
Route::get('/dashboard', function () {
    // ...
})->middleware('auth');
```

```php
// Grouper et protéger plusieurs routes
Route::middleware('auth')->group(function () {
    Route::get('/my-profile', [MyProfileController::class, 'show']);
});
```

```php
// Protéger toutes les routes d'un contrôleur
Route::resource('posts', MyProfileController::class)->middleware('auth');
```

Combiner middlewares et policies

Les middlewares et les policies peuvent être combinés pour une sécurité renforcée :

1. Le middleware `auth` s'assure que la personne est connectée.
2. La policy vérifie les permissions nécessaires.

```php
public function update(Request $request, Post $post)
{
    Gate::authorize('update', $post);

    // Continue la mise à jour du post dans le contrôleur PostController...
}
```

Conclusion

L'authentification permet de vérifier l'identité d'une personne, tandis que l'autorisation détermine ce qu'elle peut faire.

Laravel fournit des outils puissants : `Auth`, `Hash`, gates, policies et middlewares pour gérer ces aspects.

Concepts essentiels pour construire des applications web sécurisées.

Questions

Est-ce que vous avez des questions ?

À vous de jouer !

- (Re)lire le contenu de cours.
- Faire les exercices.
- Faire le mini-projet.
- Poser des questions si nécessaire.

➡️ Visualiser le contenu complet sur GitHub.

N'hésitez pas à vous entraidez si vous avez des difficultés !

Sources

- Illustration principale par Richard Jacobs sur Unsplash
- Illustration par Aline de Nadai sur Unsplash
- Illustration par CDC sur Unsplash
- Illustration par Imre Tomosvari sur Unsplash
- Illustration par Nikita Kachanovsky sur Unsplash