Laravel - Blog APP: Login - Part 2

1 - Configurer le contrôleur d'authentification

Laravel est déjà livré avec un tableau des utilisateurs et des méthodes d'authentification, de connexion, de cryptage de mot de passe, de session, etc.

Pour démarrer le processus d'authentification, nous allons d'abord créer le contrôleur : CustomAuthController

```
php artisan make:controller CustomAuthController -m User
```

Dans ce contrôleur, nous utiliserons certaines classes Laravel comme: L'authentification, le chiffrement de mot de passe(hash), le modèle utilisateur, la session.
Chargez-le en utilisant le code suivant en haut du fichier.

```
use Hash;
use Session;
use Illuminate\Support\Facades\Auth;
```

Définissez une méthode pour ce contrôleur :

index(): Pour charger la page de connexion

```
public function index()
{
   return view('auth.login');
}
```

create(): Pour charger le formulaire enregistrez un nouvel utilisateur

```
public function create()
{
   return view('auth.registration');
}
```

store(): la demande de post pour saisir les données de l'utilisateur avec un mot de passe chiffré.

```
public function store(Request $request)
{
  User::create([
    'name' => $request->name,
    'email' => $request->email,
    'password' => Hash::make($request->password)
  ]);
```

```
}
```

---

2 - Créer des routes d'authentification

L'étape suivante consiste à créer des routes avec les méthodes POST et GET pour gérer ces 3 méthodes lors de l'authentification personnalisée dans l'application laravel.

```
use App\Http\Controllers\CustomAuthController;

Route::get('login', [CustomAuthController::class, 'index']);
Route::get('registration', [CustomAuthController::class, 'create']);
Route::post('custom-registration', [CustomAuthController::class, 'store'])->name('register.custom');
```

---

3 - Créer des fichiers vue blade d'authentification - Login

Créez le dossier auth dans le dossier resources/views/ et créez également un nouveau fichier login.blade.php à l'intérieur, puis placez le code suivant dans le fichier resources/views/auth/login.blade.php :

```
@extends('layouts.app')

@section('content')
<main class="login-form">
   <div class="cotainer">
      <div class="row justify-content-center">
         <div class="col-md-4 pt-4">
            <div class="card">
               <h3 class="card-header text-center">Login</h3>
               <div class="card-body">
                  <form method="POST" action="{{ route('login.custom') }}">
                     @csrf
                     <div class="form-group mb-3">
                        <input type="text" placeholder="Email" id="email" class="form-control"
name="email" required
                           autofocus>
                        @if ($errors->has('email'))
                        <span class="text-danger">{{ $errors->first('email') }}</span>
                        @endif
                     </div>

                     <div class="form-group mb-3">
                        <input type="password" placeholder="Password" id="password" class="form-
control" name="password" required>
                        @if ($errors->has('password'))
                        <span class="text-danger">{{ $errors->first('password') }}</span>
                        @endif
                     </div>

                     <div class="form-group mb-3">
                        <div class="checkbox">
                           <label>
                              <input type="checkbox" name="remember"> Remember Me
                           </label>
                        </div>
                     </div>

                     <div class="d-grid mx-auto">
                        <button type="submit" class="btn btn-dark btn-block">Signin</button>
                     </div>
                  </form>

               </div>
            </div>
         </div>
      </div>
   </div>
</main>
@endsection
```

4 - Créer des fichiers vue blade d'authentification - Registration

Dans le dossier auth, créez un nouveau fichier registration.blade.php à l'intérieur, puis mettez à jour le code suggéré dans le fichier resources/views/auth/registration.blade.php :

```
@extends('layouts.app')

@section('content')
<main class="signup-form">
   <div class="cotainer">
      <div class="row justify-content-center">
         <div class="col-md-4 pt-4">
            <div class="card">
               <h3 class="card-header text-center">Register User</h3>
               <div class="card-body">

                  <form action="{{ route('register.custom') }}" method="POST">
                     @csrf
                     <div class="form-group mb-3">
                        <input type="text" placeholder="Name" id="name" class="form-control"
name="name"
                           required autofocus>
                        @if ($errors->has('name'))
                        <span class="text-danger">{{ $errors->first('name') }}</span>
                        @endif
                     </div>

                     <div class="form-group mb-3">
                        <input type="text" placeholder="Email" id="email_address" class="form-control"
                           name="email" required autofocus>
                        @if ($errors->has('email'))
                        <span class="text-danger">{{ $errors->first('email') }}</span>
                        @endif
                     </div>

                     <div class="form-group mb-3">
                        <input type="password" placeholder="Password" id="password" class="form-
control"
                           name="password" required>
                        @if ($errors->has('password'))
                        <span class="text-danger">{{ $errors->first('password') }}</span>
                        @endif
                     </div>

                     <div class="d-grid mx-auto">
                        <button type="submit" class="btn btn-dark btn-block">Sign up</button>
                     </div>
                  </form>

               </div>
            </div>
         </div>
      </div>
   </div>
</main>
@endsection
```

5 - Utilisation de la validation des données Laravel

Laravel propose plusieurs approches différentes pour valider les données entrantes de votre application. Par défaut, la classe du contrôleur de base de Laravel utilise un trait de validation des requêtes qui fournit une méthode pratique pour valider la requête HTTP entrante avec une variété de règles de validation puissantes.

Modifions la fonction create() pour valider les données avant de les insérer.

La liste des règles peut être trouvée sur le lien de documentation de laravel.

Référence: https://laravel.com/docs/8.x/validation#introduction

```
public function store(Request $request)
{

  $request->validate([
     'name' => 'required',
     'email' => 'required|email|unique:users',
     'password' => 'required|min:6',
  ]);

            $user = new User;
   $user->fill($request->all());
   $user->password = Hash::make($request->password);
   $user->save();

   return redirect("login");
}
```

Pour renvoyer des messages d'erreur à la vue, ouvrez la vue registration.blade.php et entrez le code ci-dessous à l'endroit où le message d'erreur doit être affiché. Ce code vérifiera si la condition de validation a renvoyé une erreur. S'il y a une erreur, le message sera stocké dans une variable de tableau appelée $errors. S'il y a une erreur, il est possible de récupérer les données précédentes.

```
@if ($errors->has('name'))

    <span class="text-danger">{{ $errors->first('name') }}</span>

@endif
```

S'il y a une erreur, il est possible de récupérer les données précédentes en insérant le code suivant dans l'input.

```
value="{{old('name')}}"
```

---

6 - Authentification de la page login

??Après avoir saisir un nouvel utilisateur, l'authentification peut être traitée. Laravel nous aide avec la classe Auth.

Il faut d'abord créer la **route**:

```
Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');
```

Notez que la route utilise un nom (->name('login.custom') qui doit être lié à l'action du formulaire de cette manière : action="{{ route('login.custom') }}".

Après la route, il faut créer la méthode d'authentification customLogin dans le **contrôleur**, avec la classe de validation.

```
public function customLogin(Request $request)
{
    $request->validate([
        'email' => 'required',
        'password' => 'required',
    ]);

    $credentials = $request->only('email', 'password');
    if (Auth::attempt($credentials)) {
        return redirect()->intended('dashboard');
    }

    return redirect("login")->withSuccess('Les informations de connexion ne sont pas valides ');
}
```

Pour récupérer le message d'erreur ajoutez le code suivant au fichier login.blade.php

```
@if(session('success'))
    <h4 class="text-danger">{{session('success')}}</h4>
@endif
```

---

7 - Créez une page de tableau de bord (dashboard).

La page du tableau de bord sera un menu d'accueil. si l'utilisateur est connecté il peut accéder à la page du blog, sinon il est invité à faire sa connexion.

Nous devons créer la route du tableau de bord :

```
Route::get('dashboard', [CustomAuthController::class, 'dashboard']);
```

La méthode dashboard() au contrôleur CustomAuthController:

```
public function dashboard()
{
    $name = null;
    if(Auth::check()){
      $name = Auth::user()->name;
    }
      return view('blog.dashboard', ['name'=> $name]);
```

```
    }
```

et la vue blog/dashboard.blade.php:

```
@extends('layouts.app')
@section('content')

  <nav class="navbar navbar-light navbar-expand-lg mb-5">
    <div class="container">
      <a class="navbar-brand mr-auto" href="#">My Blog : {{ $name }}</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav"
        aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          @guest
          <li class="nav-item">
            <a class="nav-link" href="login">Login</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="registration">Register</a>
          </li>
          @else
          <li class="nav-item">
            <a class="nav-link" href="blog">Blog</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{{ route('logout') }}">Logout</a>
          </li>
          @endguest
        </ul>
      </div>
    </div>
  </nav>

@endsection
```

---

8 - Créer la page de déconnexion

Nous devons créer la route de deconnexion:

```
Route::get('logout', [CustomAuthController::class, 'logout'])->name('logout');
```

La méthode logout() au contrôleur CustomAuthController:

```
public function logout() {
    Session::flush();
    Auth::logout();

    return Redirect('login');
}
```

---

9 - Sécuriser la page Blog

Désormais, seul un utilisateur connecté doit accéder à la page du blog, pour ce faire, nous devons travailler dans le contrôleur BlogPostController, en utilisant la classe Auth :

```
use Auth;
```

et ajoutez la condition d'accès dans la méthode index().

```
public function index()
{
  if(Auth::check()){
  $posts = BlogPost::all(); // select * from BlogPost
  return  view('blog.index', ['posts' => $posts]);
  }
  return redirect("login")->withSuccess('Vous n\'êtes pas autorisé à accéder');
}
```

---

10 - Vérifier la session de sécurité avec le middleware

Middleware

- Le middleware fournit un mécanisme pratique pour inspecter et filtrer les requêtes HTTP entrant dans votre application. Par exemple, Laravel inclut un middleware qui vérifie que l'utilisateur de votre application est authentifié. Si l'utilisateur n'est pas authentifié, le middleware redirigera l'utilisateur vers l'écran de connexion de votre application. Cependant, si l'utilisateur est authentifié, le middleware permettra à la demande de continuer dans l'application.
- Un middleware supplémentaire peut être écrit pour effectuer diverses tâches en plus de l'authentification. Par exemple, un middleware de journalisation peut journaliser toutes les demandes entrantes dans votre application. Il existe plusieurs middleware inclus dans le framework Laravel, y compris un middleware pour l'authentification et la protection CSRF. Tous ces middlewares se trouvent dans le répertoire app / Http / Middleware.

Référence: https://laravel.com/docs/8.x/middleware

Pour utiliser le middleware pour vérifier les identifiants de connexion dans notre projet de blog, ajoutez l'alias de route à la route de connexion.

```
Route::get('login', [CustomAuthController::class, 'index'])->name('login');
```

Puis ajoutez le middleware d'authentification ->middleware('auth'), à toutes les routes que vous souhaitez sécuriser.

```
Route::get('/blog',[BlogPostController::class, 'index'])->middleware('auth');
```
