BRAINSTER

Laravel / Auth

Brainster Web Development Academy

Exercise

Create a laravel project and implementation authentication options (register and login) using some of the built-in Laravel packages for that purpose (Breeze or Jetstream).

By default, Breeze doesn’t offer roles. You need to add logic to the code responsible for authentication so that different roles are supported. Start by making a migration to create a `roles` table. Make a seeder that will insert 3 roles in this table: regular, editor and admin.

Continue by adding a new column `role_id` in the users table and make it a foreign key referencing the `id` column on the `roles` table (don’t forget to add a function for rollback too).

Create 3 users using a seeder, each with different role. Don’t forget to hash their passwords. Then create a middleware that will redirect them to three different pages depending on the role they have.
The regular user goes to the regular ‘home’ page after login that shows `You are logged in!`.
The admin gets redirected to the admin page that shows `You are logged in as admin!`.
The editor gets redirected to the editor page that shows `You are logged in as editor!`.

Exercise

Now, adapt the registration form so that new users can choose whether they want to register as regular users or as editors. Admin is not an available option, admin users will be added only with seeders.

Continue by creating the views for admins and editors. When user is logged in as editor, he should see a table of all other editors on the platform. He can delete his account (only his, not others) which logs him out of the application and permanently removes him from database.

When a user is logged in as admin, he should see a list of all the users (regular, editors and admins). He can remove any of the users and he can also deactivate a particular user.

Deactivating a user means that the user can no longer log in to the application, but he still exists in the database. This can be done using 2 different approaches. The first is to add a new column that will keep info whether given user is active (i.e. is_active column, which by default is 1, but if it is 0 then user can not log in). The second way to accomplish this would be to use Laravel's soft deletes mechanism. Deactivating a user means he is `soft-deleted`. For this, new migration is needed to add the `deleted_at` column in the users table, but the authentication logic should work out of the box.