Faculty of Applied Science
Information Technology
2023-24 Fall Semester

Web Technologies
06 - Authentication and Authorization

Previous Lecture

- Eloquent Operations
- Defining Migrations
- Running Migrations
- More Examples

Contents

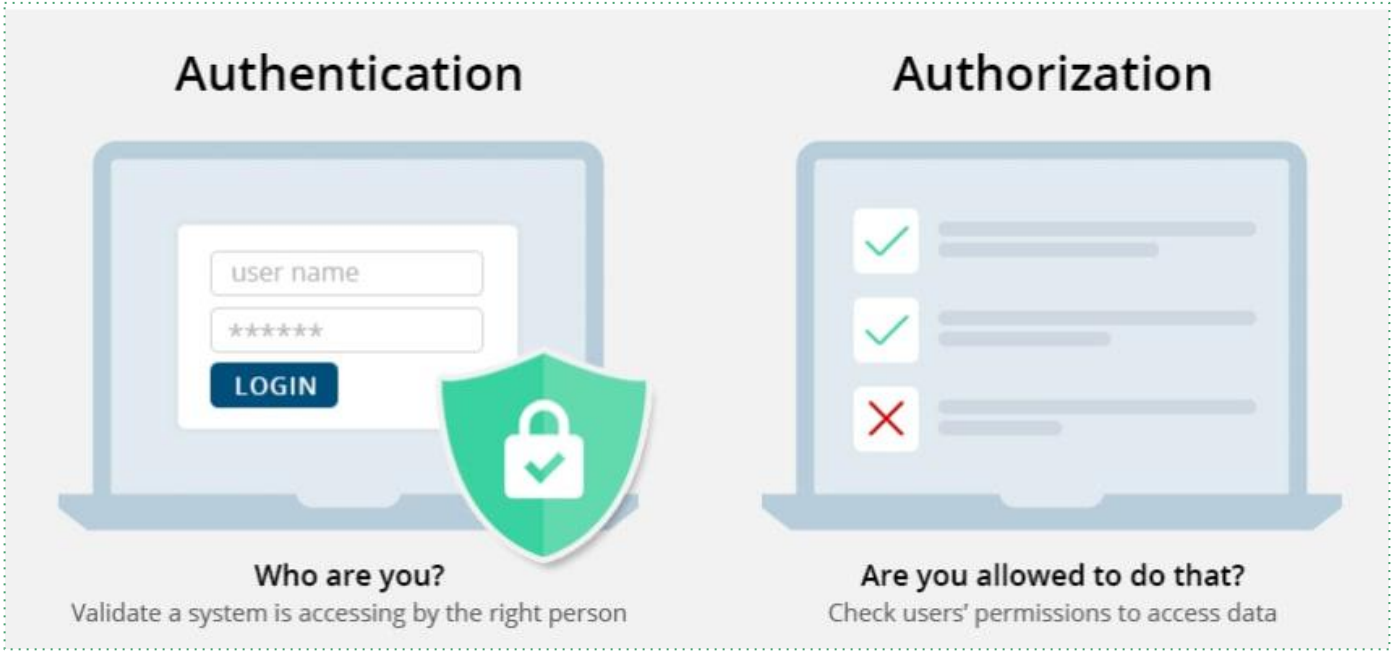
- User Authentication and Authorization
- User Model and Migration.
- Laravel Auth Package.
- Auth Controllers(Register, Login, Reset Password, Forget Password, Email Verification).
- Auth routes.

The Difference Between Authentication and Authorization

Authentication means verifying who someone is, and allowing them to act as that person in your system. This includes the login and logout processes, and any tools that allow the users to identify themselves during their time using the application.

Authorization means determining whether the authenticated user is allowed (authorized) to perform a specific behavior. For example, an authorization system allows you to forbid any non- administrators from viewing the site’s earnings.

The Difference Between Authentication and Authorization



User Authentication and Authorization

Setting up a basic user authentication system including registration, login, sessions, password resets, and access permissions can often be one of the more time-consuming pieces of creating the foundation of an application. It’s a prime candidate for extracting functionality out to a library, and there are quite a few such libraries

User Authentication and Authorization

Because of how much authentication needs vary across projects, most authentication systems grow bulky and unusable quickly. Thankfully, Laravel has found a way to make an authentication system that’s easy to use and understand, but flexible enough to fit in a variety of settings.

User Model and Migration

When you create a new Laravel application, the first migration and model you'll see are the ***create_users_table*** migration and the ***App\Models\User*** model. Example in the next slide shows, straight from the migration, the fields you'll get in your users table.

User Model and Migration

create_users_table

```
Schema::create('users', function (Blueprint $table) {
    $table->bigIncrements('id');
    $table->string('name');
    $table->string('email')->unique();
    $table->string('password');
    $table->rememberToken();
    $table->timestamps();
});
```

User Model and Migration

We have an auto incrementing primary key ID, a name, a unique email, a password, a “remember me” token, and created and modified timestamps. This covers everything you need to handle basic user authentication in most apps.

Authentication Quickstart

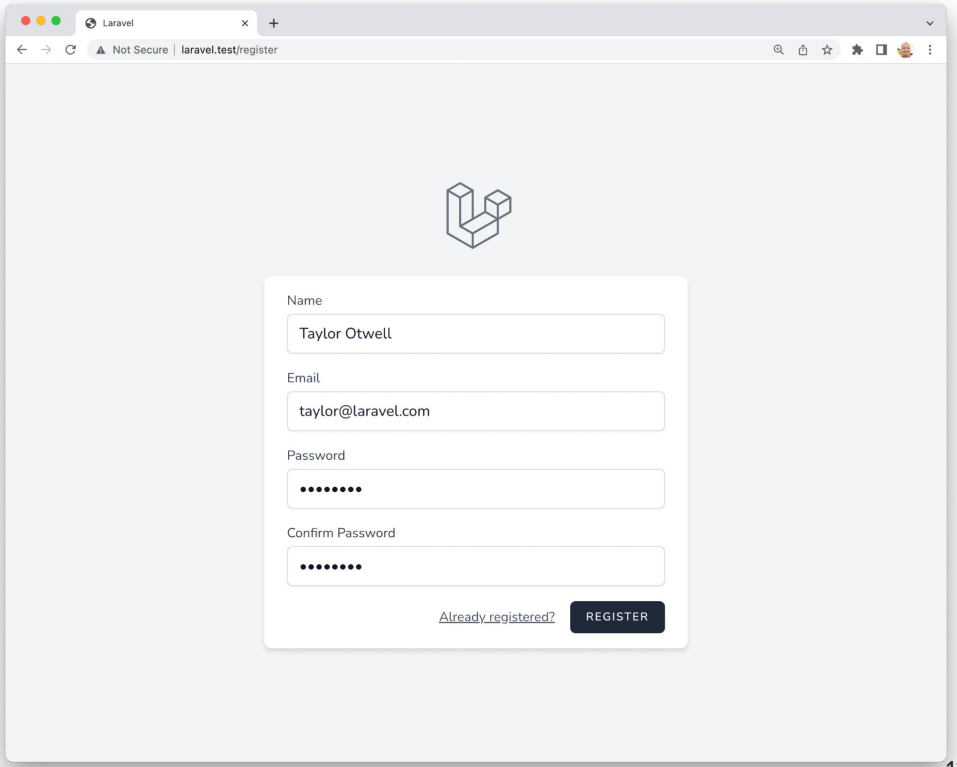
Laravel ships with several pre-built authentication controllers, which are located in the ***App\Http\Controllers\Auth*** namespace. The ***RegisterController*** handles new user registration, the LoginController handles authentication, the ***ForgotPasswordController*** handles e-mailing links for resetting passwords, and the ***ResetPasswordController*** contains the logic to reset passwords. Each of these controllers uses a trait to include their necessary methods. For many applications, you will not need to modify these controllers at all.

Authentication Starter Kits

Laravel Breeze is a minimal, simple implementation of all of Laravel's authentication features, including login, registration, password reset, email verification, and password confirmation. In addition, Breeze includes a simple "profile" page where the user may update their name, email address, and password.

Authentication Starter Kits

Laravel Breeze's default view layer is made up of simple Blade templates styled with **Tailwind** CSS. Or, Breeze can scaffold your application using Vue or React and Inertia.



Authentication Starter Kits

First, you should create a new Laravel application

- configure your database
- run your database migrations

Once you have created a new Laravel application, you may install Laravel Breeze using Composer:

```
composer require laravel/breeze --dev
```

Authentication Starter Kits

After Composer has installed the Laravel Breeze package, you may run the **breeze:install** Artisan command. This command publishes the authentication views, routes, controllers, and other resources to your application.

```
php artisan breeze:install

php artisan migrate
npm install
npm run dev
```

Authenticating

Now that you have routes and views setup for the included authentication controllers, you are ready to register and authenticate new users for your application!

you may navigate to your application's **/login** or **/register** URLs in your web browser. All of Breeze's routes are defined within the **routes/auth.php** file.

Authenticating

Retrieving The Authenticated User

You may access the authenticated user via the ***Auth*** facade:

```php
// Get the currently authenticated user...
$user = Auth::user();


// Get the currently authenticated user's ID...
$id = Auth::id();
```

Authenticating

Determining If The Current User Is Authenticated

To determine if the user is already logged into your application, you may use the ***check*** method on the ***Auth*** facade, which will return ***true*** if the user is authenticated:

```
if (Auth::check()) {
    // The user is logged in...
}
```

Protecting Routes

Route middleware can be used to only allow authenticated users to access a given route. Laravel ships with an ***auth*** middleware, which is defined at ***Illuminate\Auth\Middleware\Authenticate***. Since this middleware is already registered in your HTTP kernel, all you need to do is attach the middleware to a route definition:

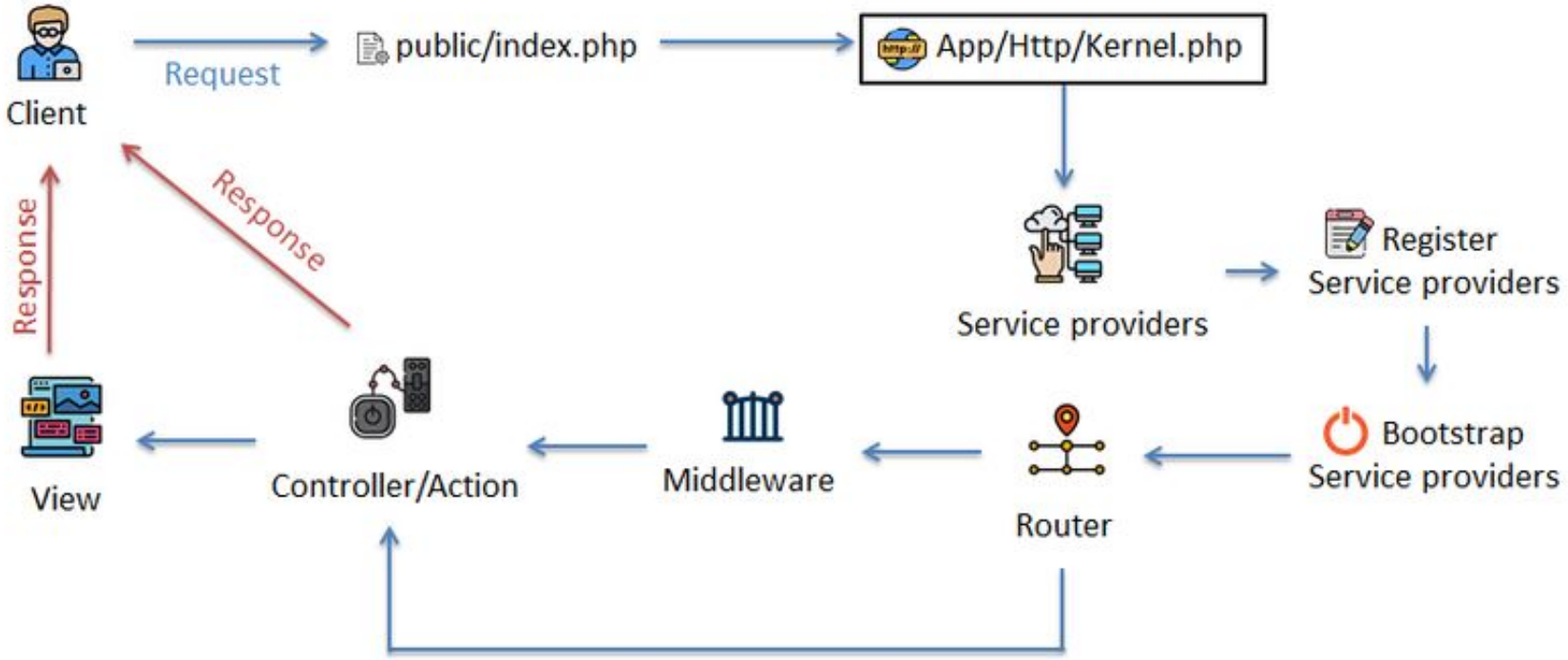
```
Route::get('profile', function () {
    // Only authenticated users may enter...
})->middleware('auth');
```

Protecting Routes

If you are using controllers, you may call the ***middleware*** method from the controller's constructor instead of attaching it in the route definition directly:

```
public function __construct()
{
    $this->middleware('auth');
}
```

Request Lifecycle (From Lecture 3)



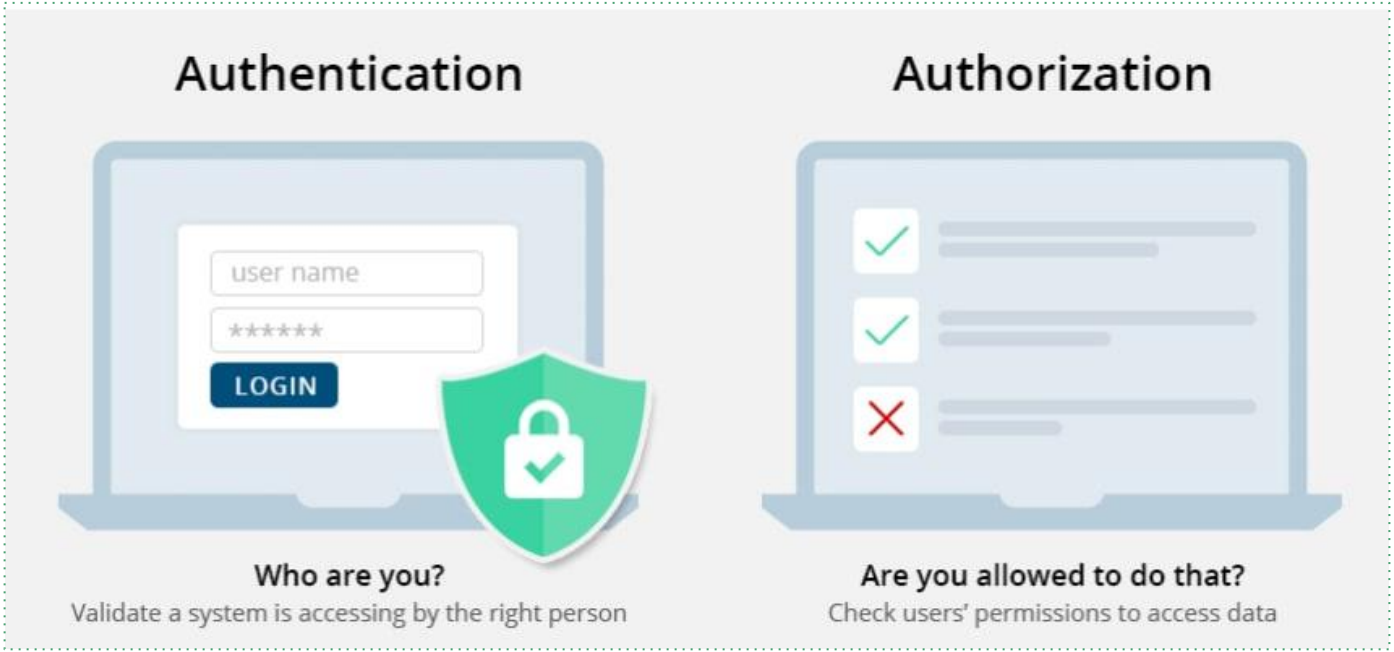
Additional Authentication Packages and kits

- **Laravel Jetstream:** While Laravel Breeze provides a simple and minimal starting point for building a Laravel application, Jetstream augments that functionality with more robust features and additional frontend technology stacks.

- **Laravel Fortify:** Laravel Fortify is a frontend agnostic authentication backend implementation for Laravel. Fortify registers the routes and controllers needed to implement all of Laravel's authentication features

- **Laravel Spark:** Laravel Spark allows you to define subscription plans for your application and provides your customers with a convenient billing portal

The Difference Between Authentication and Authorization



Authorization

In addition to providing authentication services out of the box, Laravel also provides a simple way to authorize user actions against a given resource. Like authentication, Laravel's approach to authorization is simple, and there are two primary ways of authorizing actions: **gates** and **policies**.

Authorization

Think of **gates** and **policies** like **routes** and **controllers**. Gates provide a simple, Closure based approach to authorization while policies, like controllers, group their logic around a particular model or resource. We'll explore gates first and then examine policies.

Authorization

You do not need to choose between exclusively using gates or exclusively using policies when building an application.

Most applications will most likely contain a mixture of gates and policies, and that is perfectly fine! **Gates are most applicable to actions which are not related to any model or resource**, such as viewing an administrator dashboard. In contrast, **policies should be used when you wish to authorize an action for a particular model or resource**.

Gates

Writing Gates

Gates are Closures that determine if a user is authorized to perform a given action and are typically defined in the ***App\Providers\AuthServiceProvider*** class using the Gate facade

Gates always receive a **user** instance as their first argument, and may optionally receive additional arguments such as a relevant Eloquent model.

```php
public function boot()
{
    $this->registerPolicies();

    Gate::define('edit-settings', function ($user) {
        return $user->isAdmin;
    });

    Gate::define('update-post', function ($user, $post) {
        return $user->id === $post->user_id;
    });
}
```

Gates

Authorizing Actions

To authorize an action using gates, you should use the ***allows*** or ***denies*** methods. Note that you are not required to pass the currently authenticated user to these methods. Laravel will automatically take care of passing the **user** into the gate Closure

```
if (Gate::allows('edit-settings')) {
    // The current user can edit settings
}
```

Gates

Authorizing Actions

```
if (Gate::allows('update-post', $post)) {
    // The current user can update the post...
}

if (Gate::denies('update-post', $post)) {
    // The current user can't update the post...
}
```

Gates

Authorizing Actions

If you would like to determine if a particular user is authorized to perform an action, you may use the ***forUser*** method on the Gate ***facade***:

```
if (Gate::forUser($user)->allows('update-post', $post)) {
    // The user can update the post...
}
```

Additional Authorization Packages and kits

- **Laravel-permission:** This package allows you to manage user permissions and roles in a database.Because all permissions will be registered on Laravel's gate, you can check if a user has a permission with Laravel's default can function:

Activities and Next Week Topics

This Week:

- Read the rest of Chapter 09 of Laravel: Up & Running, for more information User Authentication and Authorization.
- Try to implement these Authentication techniques on your projects.
- Prepare index(),create() and store() for one of your project models, and bring it to the lab.
- Study for the quiz.

Next Week:

- Testing: The Testing Environment, Simple Unit Tests, HTTP Tests, Database Tests.
- Git and Version Control System.

References / Further Readings

- Laravel.com : Laravel’s official Documentation.
- Matt Stauffer, 2019. Laravel: Up & Running: A Framework for Building Modern PHP Apps. O’Reilly Media.
- Dayle Rees, 2016. Laravel: Code Smart.