JavaScript & CSS Scaffolding

Introduction

While Laravel does not dictate which JavaScript or CSS pre-processors you use, it does provide a basic starting point using Bootstrap, React, and / or Vue that will be helpful for many applications. By default, Laravel uses NPM to install both of these frontend packages.

The Bootstrap and Vue scaffolding provided by Laravel is located in the `laravel/ui` Composer package, which may be installed using Composer:

```
composer require laravel/ui --dev
```

Once the `laravel/ui` package has been installed, you may install the frontend scaffolding using the `ui` Artisan command:

```
// Generate basic scaffolding...
php artisan ui vue
php artisan ui react

// Generate login / registration scaffolding...
php artisan ui vue --auth
php artisan ui react --auth
```

CSS

Laravel Mix provides a clean, expressive API over compiling SASS or Less, which are extensions of plain CSS that add variables, mixins, and other powerful features that make working with CSS much more enjoyable. In this document, we will briefly discuss CSS compilation in general; however, you should consult the full Laravel Mix documentation for more information on compiling SASS or Less.

JavaScript

Laravel does not require you to use a specific JavaScript framework or library to build your applications. In fact, you don't have to use JavaScript at all. However, Laravel does include some basic scaffolding to make it easier to get started writing modern JavaScript using the Vue library. Vue provides an expressive API for building robust JavaScript applications using components. As with CSS, we may use Laravel Mix to easily compile JavaScript components into a single, browser-ready JavaScript file.

Writing CSS

After installing the `laravel/ui` Composer package and generating the frontend scaffolding, Laravel's `package.json` file will include the `bootstrap` package to help you get started prototyping your application's frontend using Bootstrap. However, feel free to add or remove packages from the `package.json` file as needed for your own application. You are not required to use the Bootstrap framework to build your Laravel application - it is provided as a good starting point for those who choose to use it.

Before compiling your CSS, install your project's frontend dependencies using the Node package manager (NPM):

```
npm install
```

Once the dependencies have been installed using `npm install`, you can compile your SASS files to plain CSS using Laravel Mix. The `npm run dev` command will process the instructions in your `webpack.mix.js` file. Typically, your compiled CSS will be placed in the `public/css` directory:

```
npm run dev
```

The `webpack.mix.js` file included with Laravel's frontend scaffolding will compile the `resources/sass/app.scss` SASS file. This `app.scss` file imports a file of SASS variables and loads Bootstrap, which provides a good starting point for most applications. Feel free to customize the `app.scss` file however you wish or even use an entirely different pre-processor by configuring Laravel Mix.

Writing JavaScript

All of the JavaScript dependencies required by your application can be found in the `package.json` file in the project's root directory. This file is similar to a `composer.json` file except it specifies JavaScript dependencies instead of PHP dependencies. You can install these dependencies using the Node package manager (NPM):

```
npm install
```

By default, the Laravel `package.json` file includes a few packages such as `lodash` and `axios` to help you get started building your JavaScript application. Feel free to add or remove from the `package.json` file as needed for your own application.

Once the packages are installed, you can use the `npm run dev` command to compile your assets. Webpack is a module bundler for modern JavaScript applications. When you run the `npm run dev` command, Webpack will execute the instructions in your `webpack.mix.js` file:

```
npm run dev
```

By default, the Laravel `webpack.mix.js` file compiles your SASS and the `resources/js/app.js` file. Within the `app.js` file you may register your Vue components or, if you prefer a different framework, configure your own JavaScript application. Your compiled JavaScript will typically be placed in the `public/js` directory.

The `app.js` file will load the `resources/js/bootstrap.js` file which bootstraps and configures Vue, Axios, jQuery, and all other JavaScript dependencies. If you have additional JavaScript dependencies to configure, you may do so in this file.

Writing Vue Components

When using the `laravel/ui` package to scaffold your frontend, an `ExampleComponent.vue` Vue component will be placed in the `resources/js/components` directory. The `ExampleComponent.vue` file is an example of a single file Vue component which defines its JavaScript and HTML template in the same file. Single file components provide a very convenient approach to building JavaScript driven applications. The example component is registered in your `app.js` file:

```
Vue.component(
    'example-component',
    require('./components/ExampleComponent.vue').default
);
```

To use the component in your application, you may drop it into one of your HTML templates. For example, after running the `php artisan ui vue --auth` Artisan command to scaffold your application's authentication and registration screens, you could drop the component into the `home.blade.php` Blade template:

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

@section('content')
    <example-component></example-component>
@endsection
```

Remember, you should run the `npm run dev` command each time you change a Vue component. Or, you may run the `npm run watch` command to monitor and automatically recompile your components each time they are modified.

If you are interested in learning more about writing Vue components, you should read the Vue documentation, which provides a thorough, easy-to-read overview of the entire Vue framework.

Using React

If you prefer to use React to build your JavaScript application, Laravel makes it a cinch to swap the Vue scaffolding with React scaffolding:

```
composer require laravel/ui --dev

php artisan ui react

// Generate login / registration scaffolding...
php artisan ui react --auth
```

Adding Presets

Presets are "macroable", which allows you to add additional methods to the `UiCommand` class at runtime. For example, the following code adds a `nextjs` method to the `UiCommand` class. Typically, you should declare preset macros in a service provider:

```
use Laravel\Ui\UiCommand;

UiCommand::macro('nextjs', function (UiCommand $command) {
    // Scaffold your frontend...
});
```

Then, you may call the new preset via the `ui` command:

```
php artisan ui nextjs
```

Become a Laravel Partner

Laravel Partners are elite shops providing top-notch Laravel development and consulting. Each of our partners can help you craft a beautiful, well-architected project.

Our Partners

Laravel

Highlights

- Release Notes
- Getting Started
- Routing
- Blade Templates
- Authentication
- Authorization
- Artisan Console
- Database
- Eloquent ORM
- Testing

Resources

- Laracasts
- Laravel News
- Laracon
- Laracon EU
- Laracon AU
- Jobs
- Certification
- Forums

Partners

- Vehikl
- Tighten Co.
- Kirschbaum
- Byte 5
- 64Robots
- Cubet
- DevSquad
- Ideil
- Cyber-Duck
- ABOUT YOU
- Become A Partner

Ecosystem

- Vapor
- Forge
- Envoyer
- Horizon
- Lumen
- Nova
- Echo
- Valet
- Mix
- Spark
- Cashier
- Homestead
- Dusk
- Passport
- Scout
- Socialite

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable and creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in most web projects.