Course - Laravel Framework

Biên dịch Asset

Laravel Mix, một package được phát triển bởi Jeffrey Way, người tạo ra Laracasts, cung cấp một API để nhanh chóng các bước xây dựng webpack cho ứng dụng Laravel của bạn bằng cách sử dụng một số công cụ CSS và JavaScript phổ biến.

Tags: laravel compile asset, laravel

Giới thiệu

Laravel Mix, một package được phát triển bởi Jeffrey Way, người tạo ra Laracasts, cung cấp một API để nhanh chóng xác định các bước xây dựng webpack cho ứng dụng Laravel của bạn bằng cách sử dụng một số công cụ CSS và JavaScript phổ biến.

Nói cách khác, Mix giúp bạn dễ dàng biên dịch và rút gọn các tệp CSS và JavaScript của ứng dụng. Thông qua việc gọi chuỗi phương thức đơn giản, bạn có thể xác định nhanh chóng đường dẫn asset của mình. Ví dụ:

```
mix.js('resources/js/app.js', 'public/js')
  .postCss('resources/css/app.css', 'public/css');
```

Nếu bạn đã từng bối rối và choáng ngợp về việc bắt đầu với webpack và biên dịch asset, bạn sẽ thích dùng Laravel Mix. Tuy nhiên, bạn không bắt buộc phải sử dụng nó trong khi phát triển ứng dụng của mình; bạn có thể tự do sử dụng bất kỳ công cụ phân bổ asset nào mà bạn muốn, hoặc thậm chí không có công cụ nào cả.

> Nếu bạn cần bắt đầu xây dựng ứng dụng của mình với CSS Laravel và Tailwind , hãy xem một trong các bộ công cụ khởi động ứng dụng của chúng tôi.

Cài đặt và thiết lập

Cài đặt Node

Trước khi chạy Mix, trước tiên bạn phải đảm bảo rằng Node.js và NPM đã được cài đặt trên máy của bạn:

```
node -v
npm -v
```

Bạn có thể dễ dàng cài đặt phiên bản mới nhất của Node và NPM bằng các chương trình cài đặt đồ họa đơn giản từ trang web chính thức của Node. Hoặc, nếu bạn đang sử dụng Laravel Sail, bạn có thể gọi Node và NPM thông qua Sail:

```
./sail node -v
./sail npm -v
```

Cài đặt Laravel Mix

Bước còn lại duy nhất là cài đặt Laravel Mix. Trong một cài đặt mới của Laravel, bạn sẽ tìm thấy một tập tin *package.json* trong thư mục gốc của cấu trúc thư mục của bạn. Tập tin mặc định *package.json* đã bao gồm mọi thứ bạn cần để bắt đầu sử dụng Laravel Mix. Hãy coi tập tin này giống như *composer.json* tệp của bạn, ngoại trừ nó ghi ra các package Node thay vì các package PHP. Bạn có thể cài đặt các package mà nó tham chiếu bằng cách chạy:

```
npm install
```

Chạy Mix

Mix là một lớp cấu hình nằm trên webpack, vì vậy để chạy các tác vụ Mix, bạn chỉ cần thực thi một trong các tập lệnh NPM có trong tập tin *package.json* Laravel mặc định. Khi bạn chạy tập lệnh *dev* hoặc *production*, tất cả nội dung CSS và JavaScript của ứng dụng sẽ được biên dịch và đặt trong thư mục ứng dụng */public* của bạn:

```
// Run all Mix tasks...
npm run dev

// Run all Mix tasks and minify output...
npm run prod
```

Xem asset thay đổi nhanh chóng

Lệnh `npm run watch` sẽ tiếp tục chạy trong thiết bị đầu cuối của bạn và xem tất cả các tập tin CSS và JavaScript có liên quan để biết các thay đổi bên trong. Webpack sẽ tự động biên dịch lại nội dung của bạn khi phát hiện thay đổi đối với một trong các tập tin sau:

```
npm run watch
```

Webpack có thể không phát hiện được các thay đổi tập tin của bạn trong một số môi trường phát triển cục bộ. Nếu đây là trường hợp trên hệ thống của bạn, hãy xem xét sử dụng lệnh `watch-poll`:

```
npm run watch-poll
```

Làm việc với stylesheet

Tập tin ứng dụng *webpack.mix.js* của bạn là điểm nhập của bạn để tổng hợp tất cả nội dung. Hãy coi nó như một chương trình bao bọc cấu hình nhẹ xung quanh webpack. Các nhiệm vụ hỗn hợp có thể được xâu chuỗi với nhau để xác định chính xác cách mà asset của bạn sẽ được biên dịch.

CSS Tailwind

Tailwind CSS là một framework hiện đại, tiện ích nhất để xây dựng các trang web tuyệt vời mà không cần rời khỏi HTML của bạn. Hãy cùng tìm hiểu cách bắt đầu sử dụng nó trong một dự án Laravel với Laravel Mix. Trước tiên, chúng ta nên cài đặt Tailwind bằng NPM và tạo tập tin cấu hình Tailwind:

```
npm install

npm install -D tailwindcss

npx tailwindcss init
```

Lệnh `init` sẽ tạo một tập tin *tailwind.config.js*. Phần `content` của tập tin này cho phép bạn định cấu hình đường dẫn đến tất cả các mẫu HTML, thành phần JavaScript và bất kỳ tập tin nguồn nào khác có chứa tên class Tailwind để mọi class CSS không được sử dụng trong các tập tin này sẽ bị xóa khỏi CSS của môi trường product mà bạn xây dựng:

```
content: [
  './storage/framework/views/*.php',
  './resources/**/*.blade.php',
  './resources/**/*.js',
  './resources/**/*.vue',
```

```
],
```

Tiếp theo, bạn nên thêm từng "class" của Tailwind vào tập tin *resources/css/app.css* trong ứng dụng của mình:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Khi bạn đã cấu hình các class của Tailwind, bạn đã sẵn sàng cập nhật tập tin *webpack.mix.js* trong ứng dụng của mình để biên dịch CSS hỗ trợ Tailwind:

```
mix.js('resources/js/app.js', 'public/js')
  .postCss('resources/css/app.css', 'public/css', [
    require('tailwindcss'),
  ]);
```

Cuối cùng, bạn nên tham chiếu stylesheet của mình trong template bố cục chính của ứng dụng. Nhiều ứng dụng chọn lưu trữ template này tại *resources/views/layouts/app.blade.php*. Ngoài ra, hãy đảm bảo bạn thêm chế độ xem (name=viewport) đáp ứng cho mác meta nếu mác này chưa có:

```
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link href="/css/app.css" rel="stylesheet">
</head>
```

PostCSS

PostCSS là một công cụ mạnh mẽ để chuyển đổi CSS của bạn, được bao gồm trong Laravel Mix. Theo mặc định, Mix sử dụng plugin *Autoprefixer* phổ biến để tự động áp dụng tất cả các tiền tố CSS3. Tuy nhiên, bạn có thể tự do thêm bất kỳ plugin bổ sung nào phù hợp với ứng dụng của mình.

Đầu tiên, hãy cài đặt plugin mong muốn thông qua NPM và đưa nó vào mảng plugin của

bạn khi gọi phương thức `postCss` của Mix. Phương thức `postCss` chấp nhận đường dẫn đến tập tin CSS của bạn làm đối số đầu tiên và thư mục nơi tập tin đã biên dịch sẽ được đặt làm đối số thứ hai:

```
mix.postCss('resources/css/app.css', 'public/css', [
  require('postcss-custom-properties')
]);
```

Hoặc, bạn có thể thực thi `postCss` mà không cần bổ sung plugin nào để đạt được quá trình biên dịch và rút gọn CSS đơn giản:

```
mix.postCss('resources/css/app.css', 'public/css');
```

Sass

Phương thức `sass` này cho phép bạn biên dịch Sass thành CSS mà trình duyệt web có thể hiểu được. Phương thức `sass` chấp nhận đường dẫn đến tập tin Sass của bạn làm đối số đầu tiên và thư mục nơi tập tin đã biên dịch sẽ được đặt làm đối số thứ hai:

```
mix.sass('resources/sass/app.scss', 'public/css');
```

Bạn có thể biên dịch nhiều tệp Sass thành các tệp CSS tương ứng của riêng chúng và thậm chí tùy chỉnh thư mục đầu ra của CSS kết quả bằng cách gọi phương thức `sass` nhiều lần:

```
mix.sass('resources/sass/app.sass', 'public/css')
  .sass('resources/sass/admin.sass', 'public/css/admin');
```

Xử lý URL

Vì Laravel Mix được xây dựng trên nền webpack nên điều quan trọng là bạn phải hiểu một vài khái niệm về webpack. Để biên dịch CSS, webpack sẽ viết lại và tối ưu hóa bất kỳ lệnh gọi `url()` nào trong stylesheet của bạn. Mặc dù điều này ban đầu nghe có vẻ lạ, nhưng đó là một phần chức năng cực kỳ mạnh mẽ. Hãy tưởng tượng rằng chúng ta muốn biên dịch Sass bao gồm một URL tương đối với một hình ảnh:

```
.example {
  background: url('../images/example.png');
}
```

Chú ý: Các đường dẫn tuyệt đối cho bất kỳ lệnh css url() nào cũng sẽ bị loại trừ khỏi việc ghi lại URL. Ví dụ, `url('/images/thing.png')` hoặc `url('http://example.com/images/thing.png')` sẽ không bị sửa đổi.

Theo mặc định, Laravel Mix và webpack sẽ tìm *example.png*, sao chép nó vào thư mục *public/images* của bạn, sau đó viết lại `url()` trong stylesheet đã tạo của bạn. Như vậy, CSS đã biên dịch của bạn sẽ là:

```
.example {
  background: url(/images/example.png?d41d8cd98f00b204e9800998ecf8427e);
}
```

Tính năng này hữu ích đến mức có thể tái cấu trúc thư mục hiện tại của bạn mà nó có thể đã được cấu hình theo cách của bạn. Nếu đúng như vậy, bạn có thể viết tắt hàm `url()` lại như sau:

```
mix.sass('resources/sass/app.scss', 'public/css').options({
  processCssUrls: false
});
```

Với sự bổ sung này vào tập tin *webpack.mix.js* của bạn, Mix sẽ không còn bất kỳ sự so sánh url() hoặc bản sao chép asset nào với thư mục */public* của bạn. Nói dễ hiểu là CSS được biên dịch sẽ giống như lúc bạn tạo nó ban đầu:

```
.example {
  background: url("../images/thing.png");
}
```

Source Maps

Mặc dù bị tắt theo mặc định, source maps có thể được kích hoạt bằng cách gọi phương thức

`mix.sourceMaps()` trong tập tin *webpack.mix.js* của bạn. Mặc dù nó đi kèm với tỉ lệ tiêu hao giữa biên dịch/hiệu suất, Nhưng nó sẽ cung cấp thêm thông tin gỡ lỗi cho các công cụ dành cho developer của trình duyệt khi sử dụng asset đã biên dịch:

```
mix.js('resources/js/app.js', 'public/js')
   .sourceMaps();
```

Kiểu của Source Mapping

Webpack cung cấp nhiều kiểu source mapping khác nhau . Theo mặc định, kiểu source mapping của Mix được đặt thành `eval-source-map`, nó sẽ cung cấp thời gian re-build nhanh chóng hơn. Nếu bạn muốn thay đổi kiểu mapping, bạn có thể làm tương tự bằng phương thức `sourceMaps`:

```
let productionSourceMaps = false;

mix.js('resources/js/app.js', 'public/js')
   .sourceMaps(productionSourceMaps, 'source-map');
```

Làm việc với Javascript

Mix cung cấp một số tính năng để giúp bạn làm việc với các tập tin JavaScript của mình, chẳng hạn như biên dịch ECMAScript hiện đại, module package, rút gọn và nối các tập tin JavaScript thuần túy. Thậm chí tốt hơn, tất cả điều này hoạt động liền mạch mà không yêu cầu điều chỉnh bất kỳ cấu hình nào:

```
mix.js('resources/js/app.js', 'public/js');
```

Với một dòng mã này, bây giờ bạn có thể tận dụng lợi thế của:

- Cú pháp EcmaScript mới nhất.
- Module
- Tối giản cho thành phẩm.

Framework VueJS

Mix sẽ tự động cài đặt các plugin Babel cần thiết để hỗ trợ biên dịch tập tin component Vue khi sử dụng phương pháp `vue`. Không cần cấu hình bổ sung nào:

```
mix.js('resources/js/app.js', 'public/js')
   .vue();
```

Khi JavaScript của bạn đã được biên dịch, bạn có thể liên kết với nó trong ứng dụng của mình:

```
<head>
    <!-- ... -->

    <script src="/js/app.js"></script>
</head>
```

Thư viện React

Mix có thể tự động cài đặt các plugin Babel cần thiết để hỗ trợ thư viện React. Để bắt đầu, hãy thêm một câu lệnh gọi vào phương thức `react`:

```
mix.js('resources/js/app.jsx', 'public/js')
   .react();
```

Bên trong tiến trình, Mix sẽ tải xuống và đưa vào plugin Babel `babel-preset-react` thích hợp. Khi JavaScript của bạn đã được biên dịch, bạn có thể liên kết với nó trong ứng dụng của mình:

```
<head>
    <!-- ... -->

    <script src="/js/app.js"></script>
</head>
```

Khai thác Vendor

Một nhược điểm tiềm ẩn của việc gộp tất cả JavaScript dành riêng cho ứng dụng của bạn với các thư viện của vendor như React và Vue là nó làm cho bộ nhớ đệm dài hạn trở nên khó khăn hơn. Ví dụ: một bản cập nhật duy nhất cho code ứng dụng của bạn cũng sẽ buộc trình duyệt tải xuống lại tất cả các thư viện của vendor ngay cả khi chúng không thay đổi.

Nếu bạn có ý định cập nhật thường xuyên JavaScript của ứng dụng của mình, bạn nên xem xét việc trích xuất tất cả các thư viện của vendor thành tập tin riêng của họ. Bằng cách này, một thay đổi nào đó trong code ứng dụng của bạn cũng sẽ không ảnh hưởng đến bộ nhớ đệm của tập tin vendor.js của bạn. Phương thức extract của Mix sẽ làm cho điều này trở nên dễ dàng hơn:

```
mix.js('resources/js/app.js', 'public/js')
   .extract(['vue'])
```

Phương thức `extract` chấp nhận một mảng có tất cả các thư viện hoặc module mà bạn muốn trích xuất vào một tập tin vendor.js. Áp dụng đoạn code ở trên, Mix sẽ tạo các tập tin sau:

- *public/js/manifest.js*: Thời gian chạy tập tin cấu hình Webpack
- *public/js/vendor.js*: Thư viện vendor của bạn
- *public/js/app.js*: Code ứng dụng của bạn

Để tránh lỗi JavaScript, hãy đảm bảo tải các tập tin này theo thứ tự thích hợp:

```
<script src="/js/manifest.js"></script>
<script src="/js/vendor.js"></script>
<script src="/js/app.js"></script>
```

Điều chỉnh cấu hình webpack

Đôi khi, bạn có thể cần phải sửa đổi cấu hình Webpack bên dưới theo cách thủ công. Ví dụ: bạn có thể có một loader hoặc plugin đặc biệt cần được tham chiếu.

Mix cung cấp một phương thức `webpackConfig` hữu ích cho phép bạn hợp nhất bất kỳ cấu hình ghi đè Webpack ngắn nào. Điều này đặc biệt hấp dẫn, vì nó không yêu cầu bạn sao chép và duy trì bản sao tập tin *webpack.config.js* của riêng bạn. Phương thức `webpackConfig` chấp nhận một đối tượng, đối tượng này phải chứa bất kỳ cấu hình Webpack cụ thể nào mà bạn muốn áp dụng.

```
mix.webpackConfig({
  resolve: {
    modules: [
      path.resolve(__dirname, 'vendor/laravel/spark/resources/assets/js')
    ]
  }
});
```

Versioning/Cache Busting

Nhiều develop gắn các asset đã biên dịch của họ với dấu thời gian hoặc mã token thống nhất để buộc các trình duyệt tải asset mới nhất thay vì đưa ra các bản sao cũ của code. Mix có thể tự động xử lý điều này cho bạn bằng cách sử dụng phương thức `version`.

Phương thức `version` sẽ thêm một dữ kiện băm thống nhất vào tên tập tin của tất cả các tập tin đã biên dịch, cho phép truy xuất bộ nhớ cache thuận tiện hơn:

```
mix.js('resources/js/app.js', 'public/js')
  .version();
```

Sau khi tạo cho tập tin được một mác phiên bản, bạn sẽ không biết tên tập tin chính xác. Vì vậy, bạn nên sử dụng hàm `mix` của Laravel trong template của mình để tải asset đã được băm. Hàm `mix` sẽ tự động xác định tên hiện tại của tập tin đã bị băm này cho bạn:

```
<script src="{{ mix('/js/app.js') }}"></script>
```

Vì các tập tin được đánh mác phiên bản thường không cần thiết trong quá trình develop, nên bạn có thể cho quy trình tạo phiên bản biết chỉ cần chạy trong thành phẩm `npm run prod`:

```
mix.js('resources/js/app.js', 'public/js');

if (mix.inProduction()) {
  mix.version();
}
```

Điều chỉnh URL hỗn hợp

Nếu phần tử đã biên dịch Mix của bạn được triển khai cho một CDN (Content Delivery Network) tách biệt với ứng dụng của bạn, bạn sẽ cần thay đổi URL hỗn hợp được tạo bởi hàm `mix`. Bạn có thể thực hiện bằng cách thêm tùy chọn cấu hình `mix_url` vào tập tin cấu hình `config/app.php` của ứng dụng:

```
'mix_url' => env('MIX_ASSET_URL', null)
```

Sau khi cài đặt cấu hình URL hỗn hợp, hàm `mix` sẽ đặt tiền tố cho URL đã cấu hình khi tạo URL cho asset:

```
https://cdn.example.com/js/app.js?id=1964becbdd96414518cd
```

Tải đồng bộ hóa BrowserSync

BrowserSync có thể tự động theo dõi các tập tin của bạn để tìm các thay đổi và đưa các thay đổi của bạn vào trình duyệt mà không yêu cầu làm mới một cách thủ công. Bạn có thể kích hoạt sự hỗ trợ này bằng cách gọi phương thức `mix.browserSync()`:

```
mix.browserSync('laravel.test');
```

Tùy chọn *BrowserSync* có thể được chỉ định bằng cách truyền một đối tượng JavaScript vào phương thức `browserSync`:

```
mix.browserSync({
  proxy: 'laravel.test'
});
```

Tiếp theo, khởi động máy chủ của webpack bằng lệnh `npm run watch`. Bây giờ, khi bạn sửa đổi tập lệnh hoặc tập tin PHP nào, thì bạn cũng có thể thấy trình duyệt làm mới trang ngay lập tức để phản ánh các thay đổi của bạn.

Các biến môi trường

Bạn có thể đưa các biến môi trường vào tập lệnh *webpack.mix.js* của mình bằng cách bổ sung tiền tố vào một trong các biến môi trường trong tập tin *.env* của mình bằng `MIX_`:

```
MIX_SENTRY_DSN_PUBLIC=http://example.com
```

Sau khi biến đã được xác định trong tập tin `.env` của bạn, bạn có thể truy cập nó thông qua đối tượng `process.env`. Tuy nhiên, bạn sẽ cần phải khởi động lại tác vụ nếu giá trị của biến môi trường thay đổi trong khi tác vụ đang chạy:

```
process.env.MIX_SENTRY_DSN_PUBLIC
```

Thông báo

Khi có sẵn, Mix sẽ tự động hiển thị thông báo dưới hệ điều hành khi biên dịch, nó cung cấp cho bạn phản hồi tức thì về việc biên dịch có thành công hay không. Tuy nhiên, có thể có những trường hợp bạn muốn tắt các thông báo này. Thông báo có thể đang kích hoạt trên máy chủ của bạn. Và thông báo cũng có thể bị vô hiệu hóa bằng cách sử dụng phương thức `disableNotifications`:

```
mix.disableNotifications();
```