サーバーサイドスクリプトⅡ（Laravel）

1. 環境構築

Laravelとは、2012年にリリースされた、PHPフレームワークの中では比較的後発のフレームワークです。この授業では、Laravelを用いてWebアプリケーションを作成する基礎を学びます。

Laravelについては、別資料で説明しますが、ここではまず、Laravelを使うための環境構築を行っていきます。

内容

[サーバーサイドスクリプトⅡ（Laravel） 1](#_Toc160786192)

[０． 環境構築 1](#_Toc160786193)

[０－１．前提条件 2](#_Toc160786194)

[０－２．Composerインストール（2024.2現在 v2.7.1） 2](#_Toc160786195)

[０－３．Node.js インストール（2024.2現在 v20.11.1） 6](#_Toc160786196)

[０－４．Laravelプロジェクト作成 10](#_Toc160786197)

[０－５．Laravel Breezeの導入 15](#_Toc160786198)

[０－６．VS Code拡張機能の導入 19](#_Toc160786199)

[０－７．IDE helperの導入① 19](#_Toc160786200)

[０－８．DB接続の設定とIDE helperの導入② 22](#_Toc160786201)

[０－９．Tailwindの設定 26](#_Toc160786202)

[環境構築、および初期設定は以上です。 30](#_Toc160786203)

０－１．前提条件

前提条件として、下記の環境が揃っていること。

※1年の授業で環境構築済み。

■Apache2.4

■PHP8.1.x

■MySQL8.0.x

０－２．Composerインストール（2024.2現在 v2.7.1）

Composerというパッケージ管理システムを使用します。

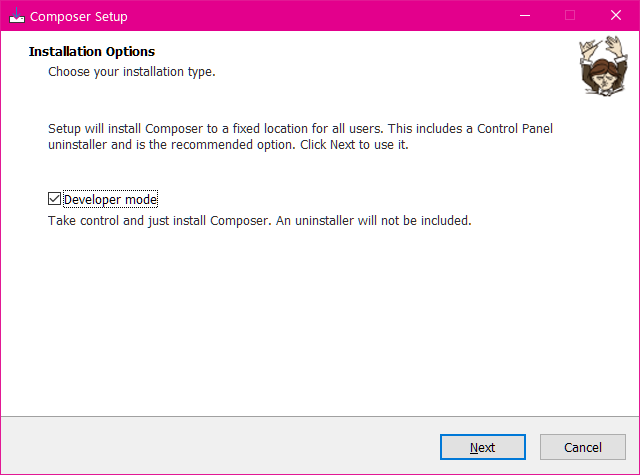
Composerは、PHPのプロジェクトにおいて、ライブラリの依存関係を管理してくれます。

■ダウンロードURL

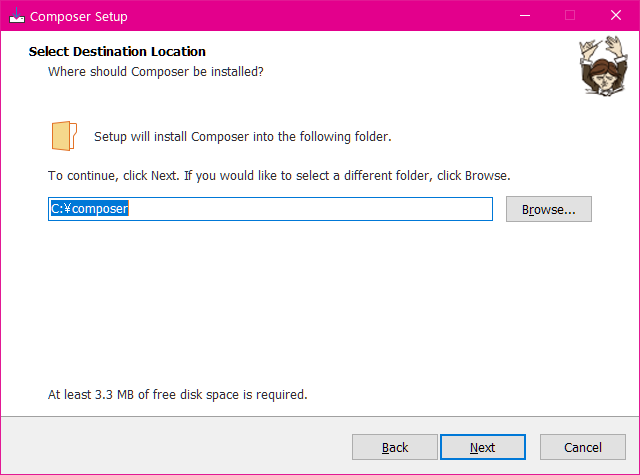
<https://getcomposer.org/doc/00-intro.md#installation-windows>

ダウンロードした[Composer-Setup.exe]を実行。

1. [Developer Mode]のチェックボックスをオンにし、[Next]押下。

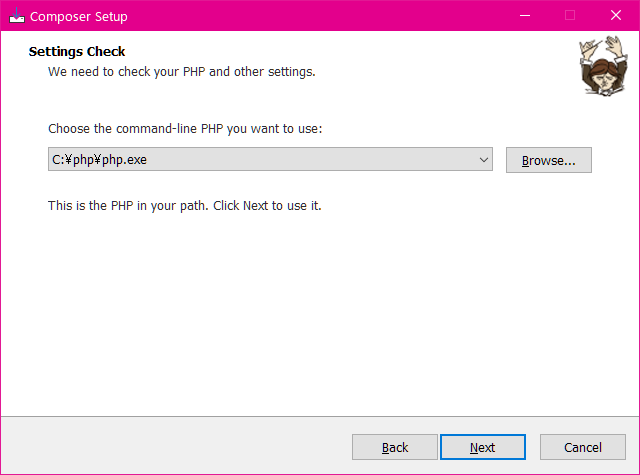


1. インストール先フォルダが [C:\composer]となっていることを確認し、[Next] 押下。

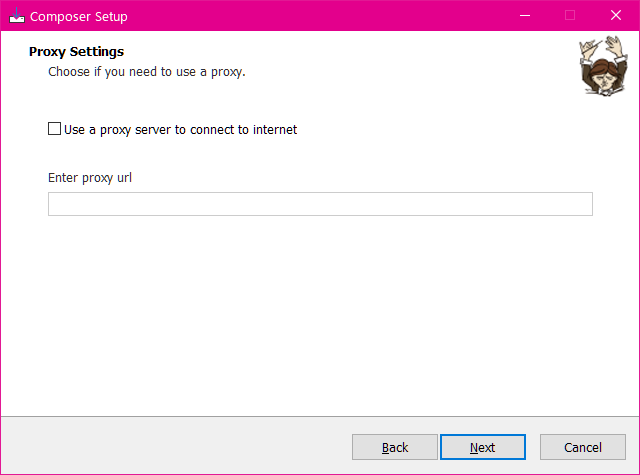


1. PHP実行ファイルが [C:\php\php.exe]となっていることを確認し、[Next] 押下。

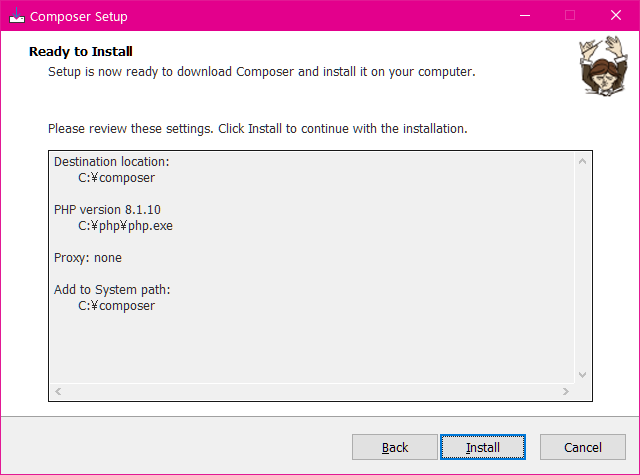
※PHP１の授業でphp実行エンジンをインストールした場所にあるexeファイルとなっていること。



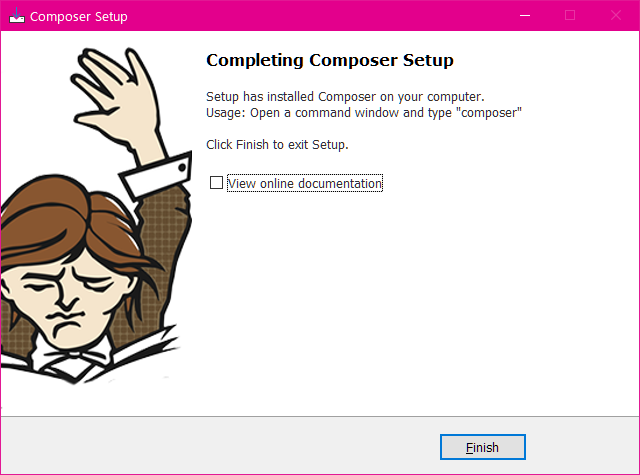
1. プロキシ設定はせず、[Next] 押下。



1. 各種設定を最終確認し、OKであれば、[Install] 押下。

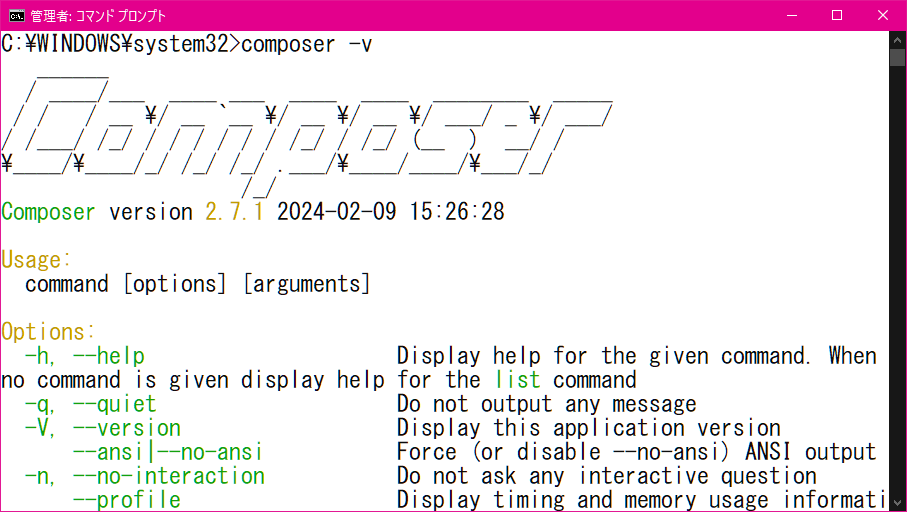


1. 完了画面が表示されたら、[Finish] 押下。



1. コマンドプロンプトを管理者で開き、「composer -v」と入力。

※下記のように、Composerのバージョンが表示されればOK。



０－３．Node.js インストール（2024.2現在 v20.11.1）

Laravel10では、Vite（ヴィート）というツールを使用します。Viteを動かすためには、Node.jsのバージョン16以上が必要となります。この項では、Node.jsのインストールを行います。

■Viteとは

Vite は、非常に高速な開発環境を提供し、実稼働用のコードをバンドルする最新のフロントエンド ビルド ツールです。Laravel でアプリケーションを構築する場合、通常は Vite を使用して、アプリケーションの CSS ファイルと JavaScript ファイルを本番環境に対応したアセットにバンドルします。

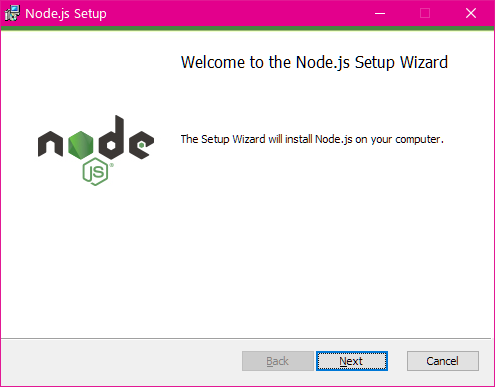
（引用元：<https://laravel.com/docs/10.x/vite#introduction>）

■ダウンロードURL

<https://nodejs.org/en/download/>

ダウンロードした[node-v20.11.1-x64.msi]を実行。

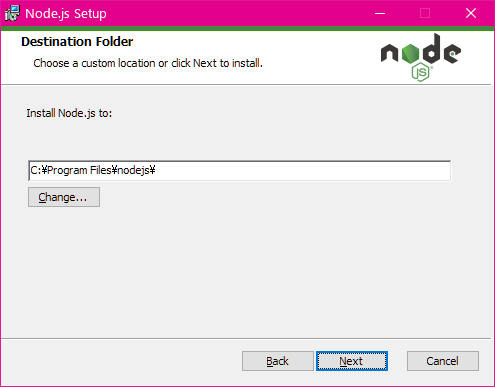
1. インストーラーが起動したら、[Next]押下。



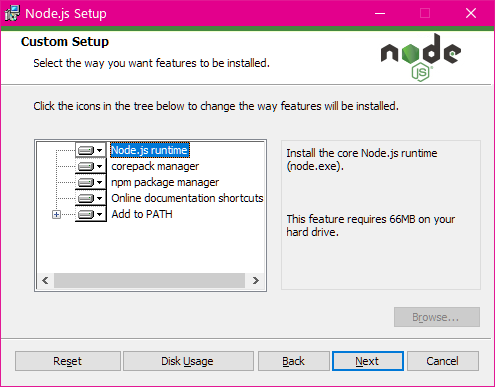
1. 承諾に関するチェックボックスをオンにし、 [Next]押下。



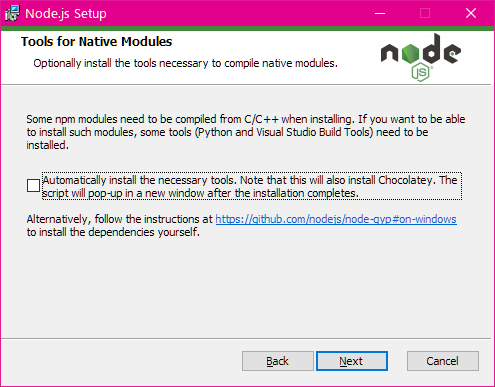
1. インストール先フォルダが[C:\Program Files\nodejs\]となっていることを確認し、 [Next]押下。



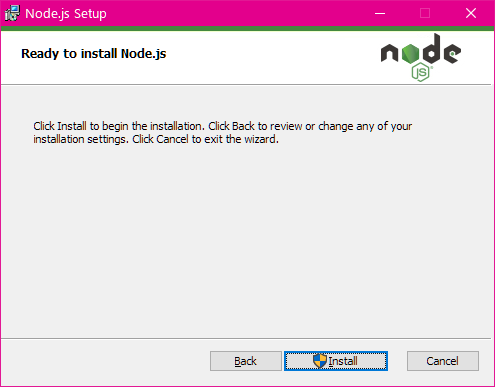
1. カスタム画面は何も変更せず、 [Next]押下。



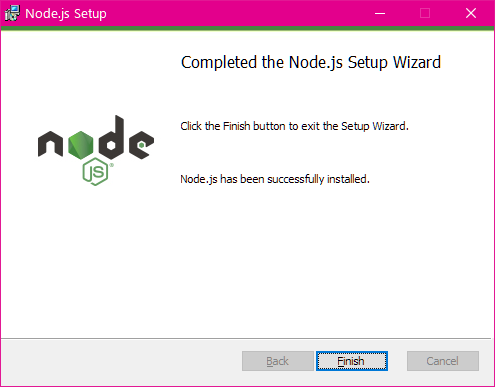
1. チェックボックスはオフのまま、 [Next]押下。



1. [Install]押下。



1. インストール完了画面が表示されたら、[Finish]押下。



1. コマンドプロンプトを管理者で開き、「node -v」と入力。

※下記のように、nodeのバージョンが表示されればOK。

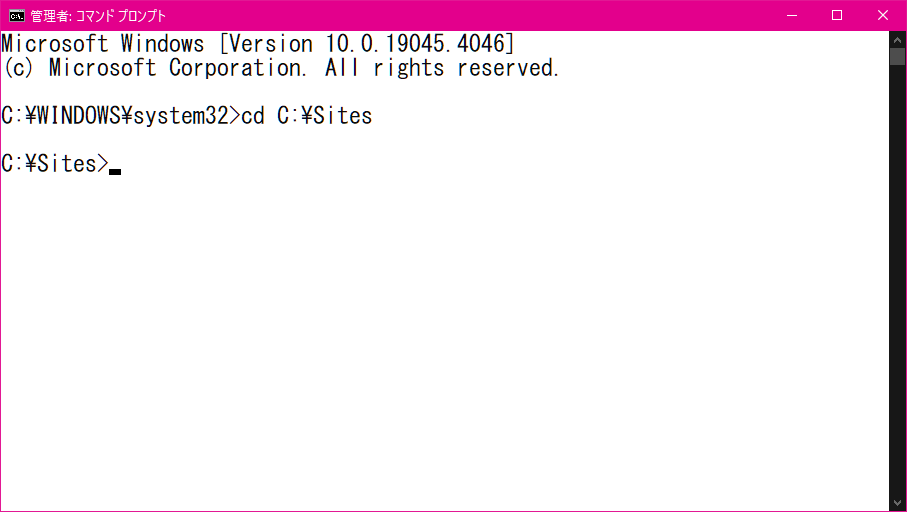


０－４．Laravelプロジェクト作成

**＊授業では、Laravel10を使用します。**

1. コマンドプロンプトを管理者で開き、[ｃｄ C:\Sites]と入し、Siteｓフォルダまで移動。

※コマンド上に[C:\Sites>]が表示されればOK。



1. コマンドで

[composer create-project --prefer-dist laravel/Laravel laravel10 “10.\*”]

と入力。

**Enterキーを押す前に、下記説明文を一読のこと。（コマンドのキャプチャのところまで）**

■基本構文（色付き文字は個別設定する箇所）

composer create-project laravel/laravel プロジェクト名 “バージョン”

※バージョンは実行時の最新のlaravelで良ければ、バージョンは省略可。（授業では指定）

■授業内使用のコマンド

composer create-project --prefer-dist laravel/laravel laravel10 “10.\*”

※「--prefer-dist」

→オプション。Composerでは「source」と「dist」でパッケージをインストール可能。

今回はダウンロードサイズが小さくインストールが速い方で実施。

※「laravel10」

　→今回作成する「プロジェクト名」

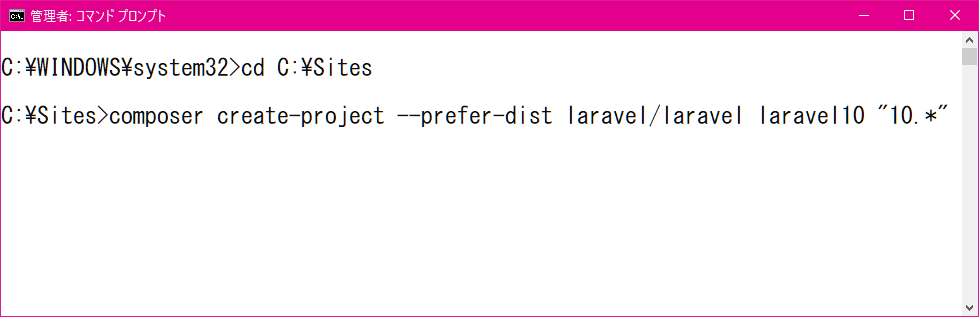
※「”**10.\*”**」

→laravel１０使用。マイナーバージョンは「.\*」にて記載。

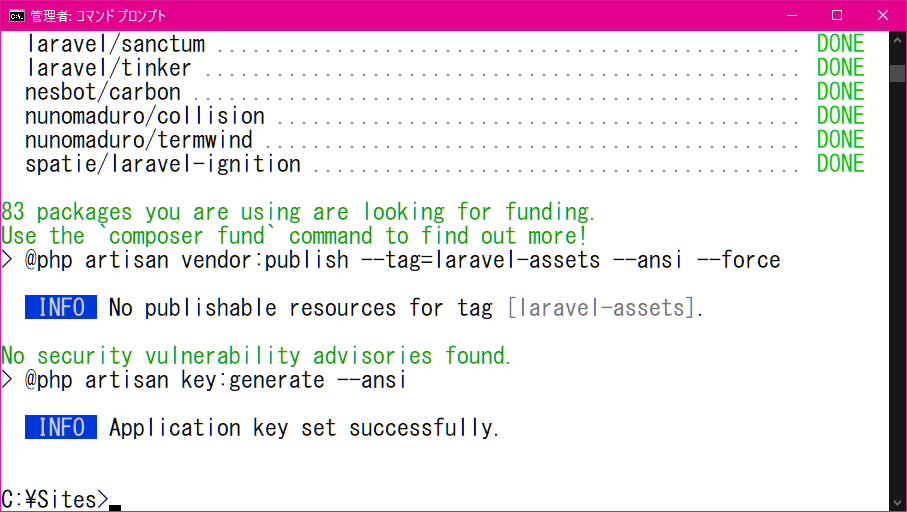
ダブルクォーテーションで囲むこと。

　　　laravelの最新版で良ければ、省略可能。

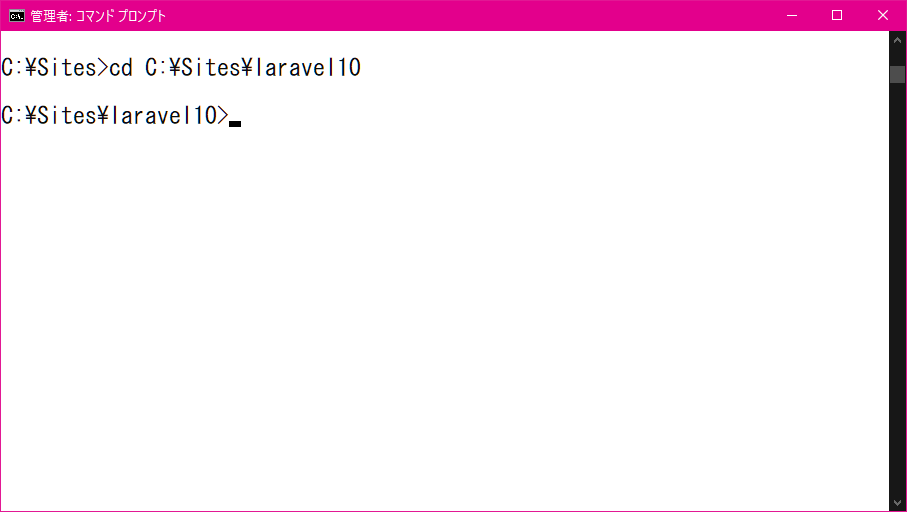
**入力したコマンドを再確認してから、Enterキー押下。**

****

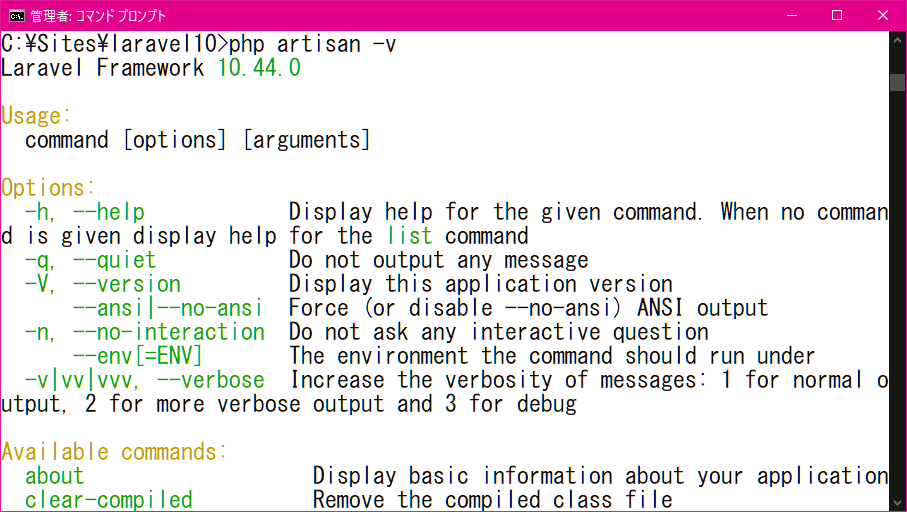
1. [～successfully]と表示され、コマンド入力可能な状態になればOK。



1. コマンドで[cd C:\Sites\laravel10]と入力し、作成したプロジェクトフォルダに移動。

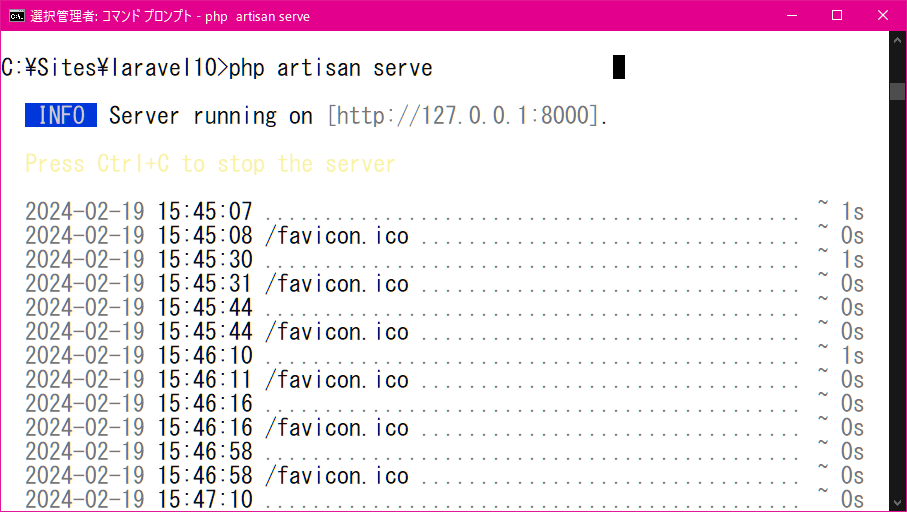


1. コマンドで[php artisan -v]と入力し、作成したプロジェクトのlaravelバージョンを確認。



1. コマンドで[php artisan serve]と入力し、laravelの仮想Webサーバーを起動。

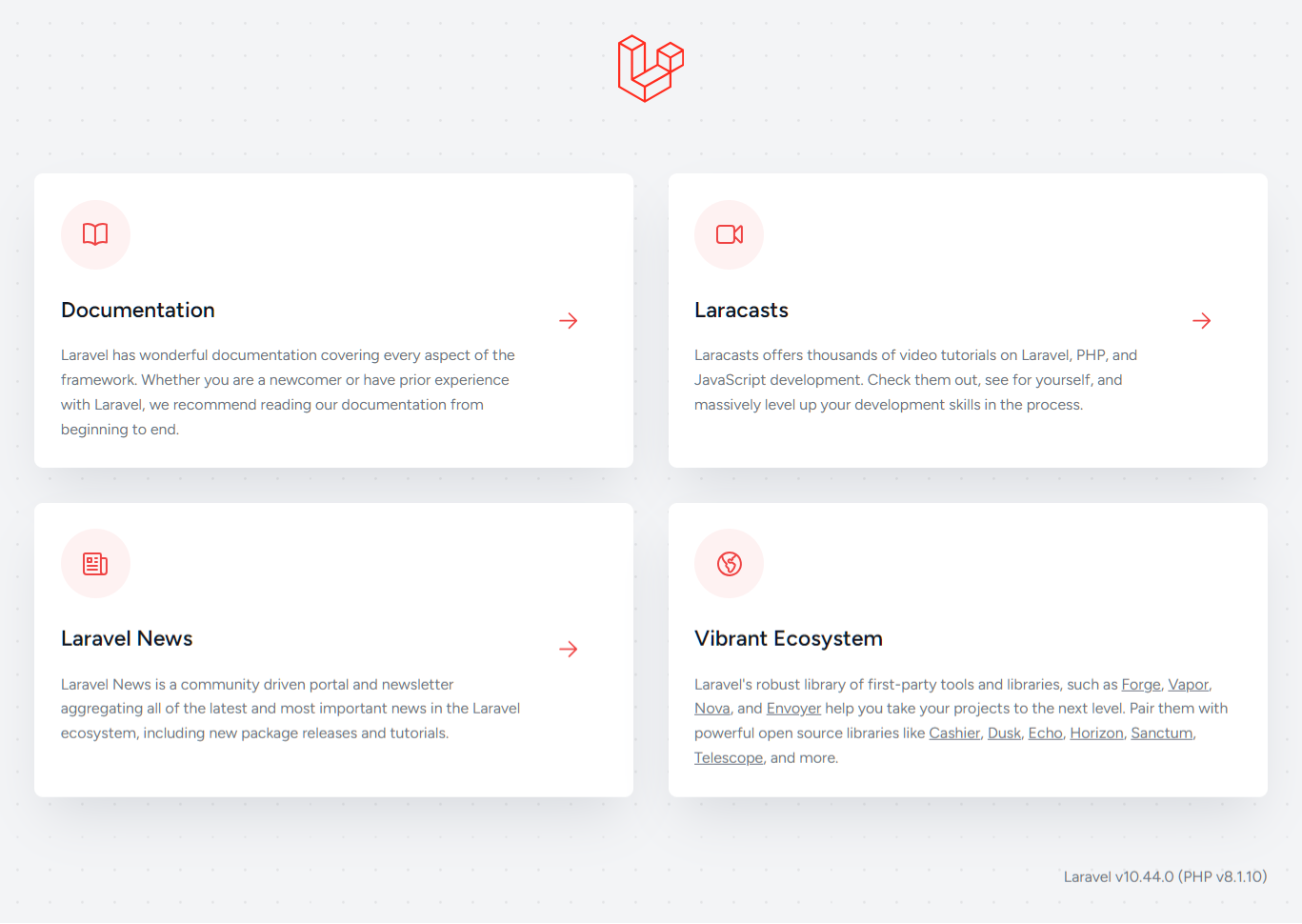
※「Server running on [http://127.0.0.1:8000]」と表示されればOK。



1. ブラウザでWelcomeページを確認。

※URLは起動時に表示の「<http://127.0.0.1:8000>」

※背景色はWindowsの色設定に依存。下図は「ライト」設定時。



1. 仮想Webサーバーを停止。コマンド上で[Ctrl]キー＋[C]キーを押下。

※コマンドプロンプトは開いたまま、次項「１－５」の手順へ。

０－５．Laravel Breezeの導入

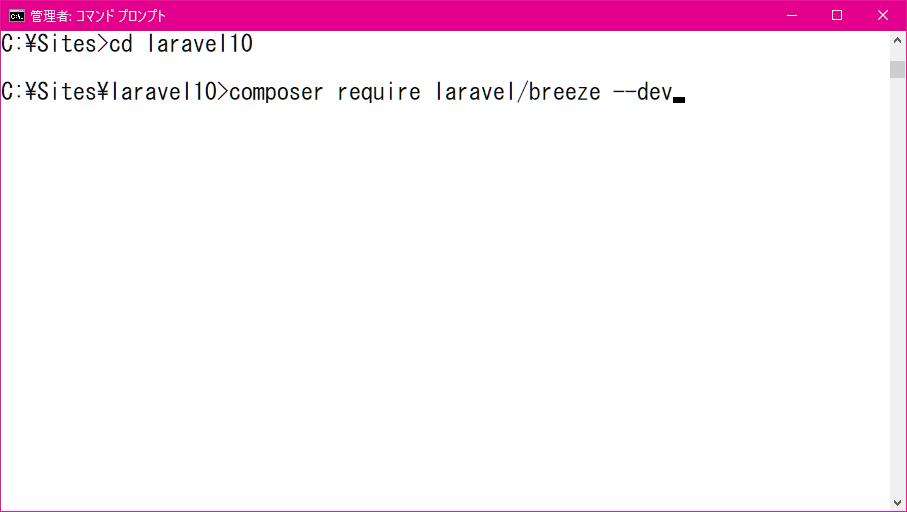
認証機能が簡単に実装できるLaravel Breezeを導入します。また、Laravel BreezeではCSSフレームワークにTailwindが使われています。授業ではこのTailwind CSSを利用します。本項では、Ｌａｒａｖｅｌ　BreezeおよびTailwindの設定方法を説明します。

1. コマンドプロンプトで現在のディレクトリが[C:\Sites\laravel10]となっていることを確認。

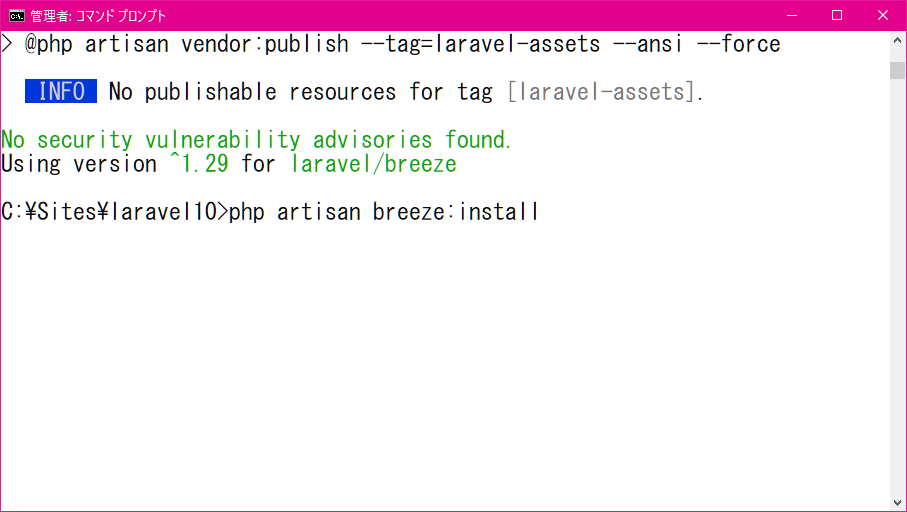
なってなければ、移動すること。

OKであれば、[composer require laravel/breeze --dev]と入力。

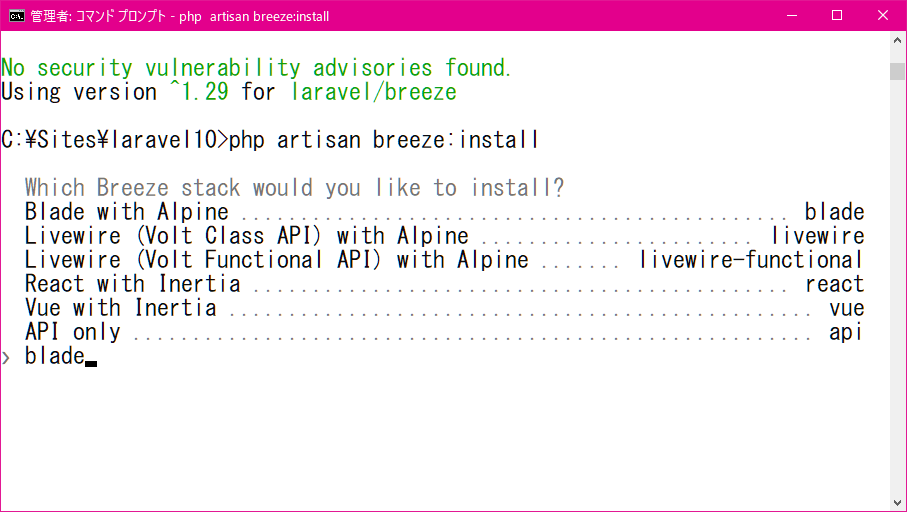
※パッケージのインストールが完了すると、コマンド上で入力可能な状態となる。



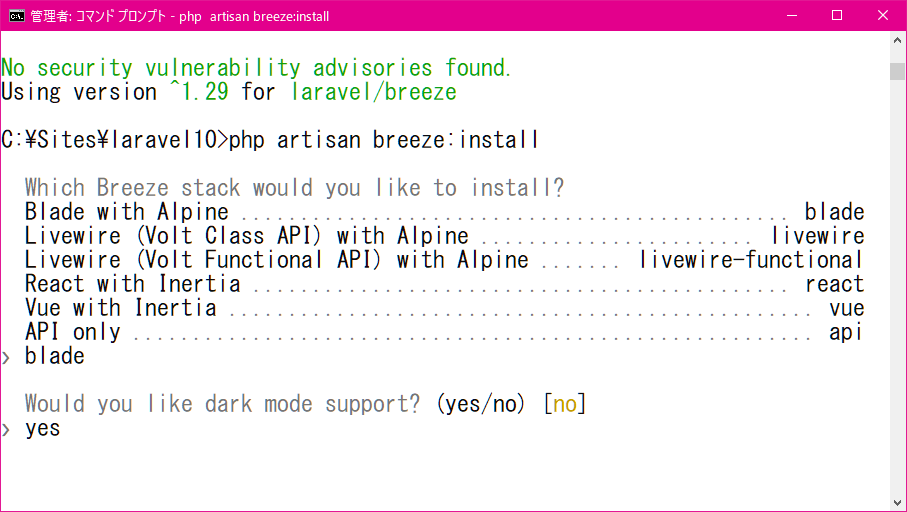
1. コマンドで[php artisan breeze:install]と入力。



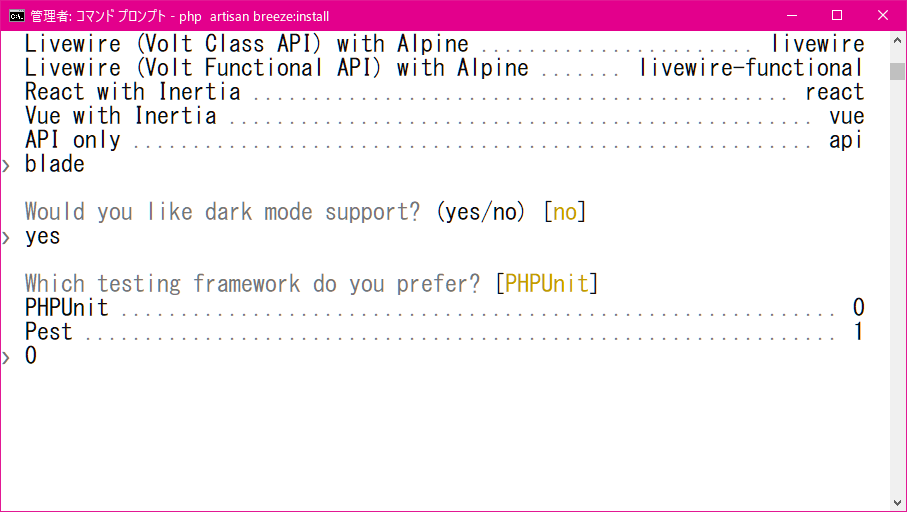
1. どのライブラリを使用するか聞かれるので、[blade]と入力。



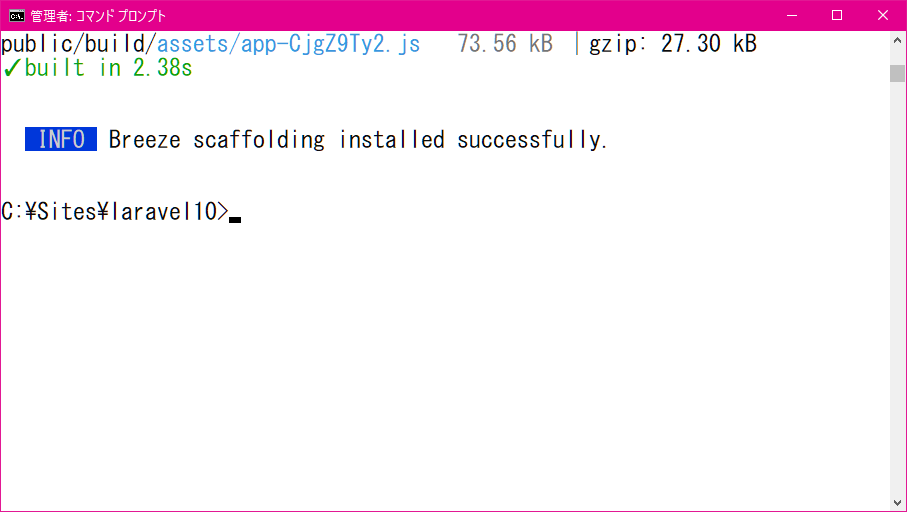
1. ダークモードをサポートするか聞かれるので、 [yes] と入力。



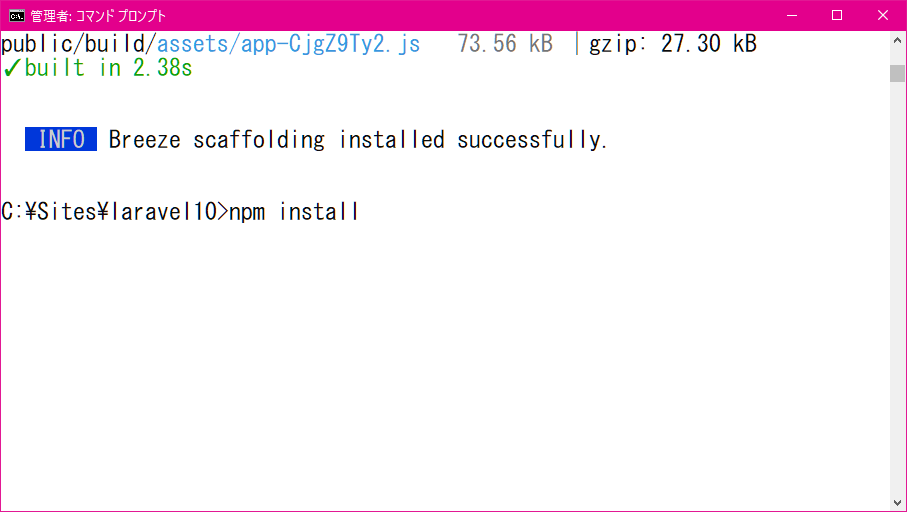
1. テストフレームワークとしてどちらにするか聞かれるので、[0] と入力。



1. Breezeについて[～successfully]と表示されていることを確認。



1. コマンドで[npm install]と入力。

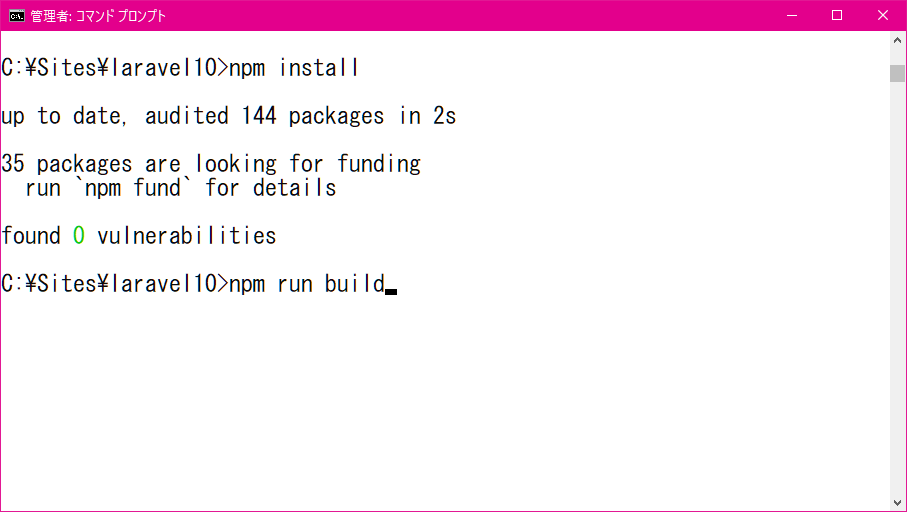


1. コマンドで[npm run build]と入力。

※[npm run dev]というコマンドもあります。

[npm run dev]だとvite開発サーバーが起動します。（開発時はこちらが便利）

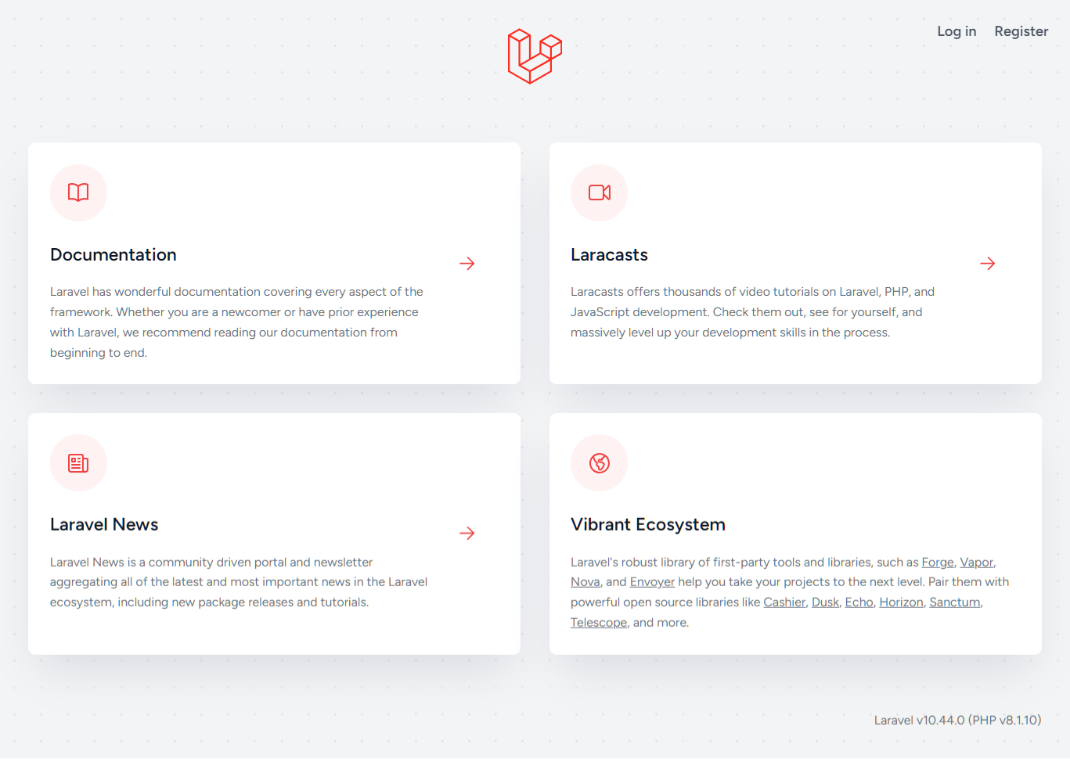
今回は、[npm run build]で実働用にアセットをバンドルし、バージョン付けしています。



1. コマンドで[php artisan serve]を入力し、仮想Webサーバーを起動。

ブラウザで [http://localhost:8000] を入力。

※認証画面が追加されていればOK。

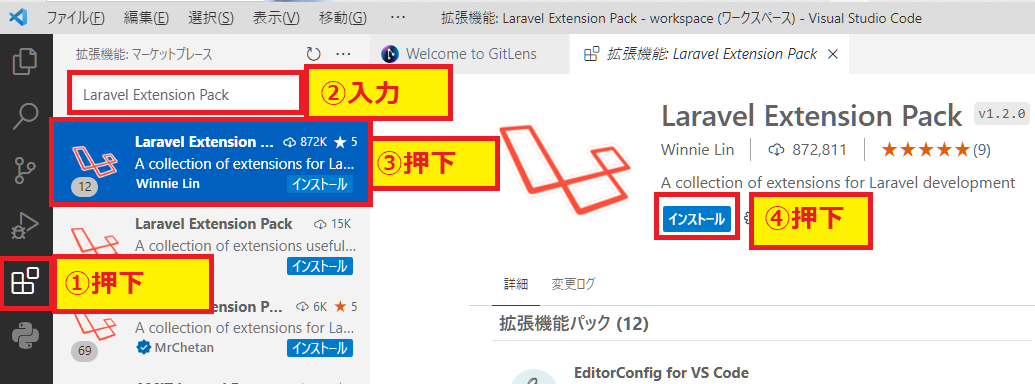


０－６．VS Code拡張機能の導入

Visual Studio Code（以降VSCodeと略）の拡張機能の導入を行います。

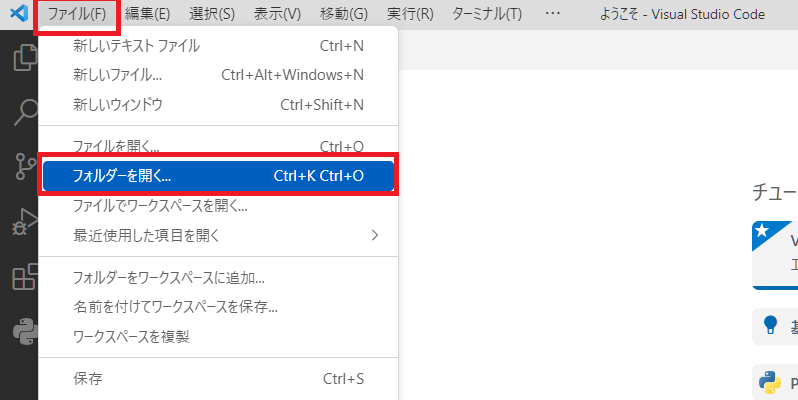
1. VSCodeを起動。下図の手順で、拡張機能から「Laravel Extension Pack」を検索して、インストール。

※Laravelでの開発をサポートしてくれるいくつもの拡張機能がまとめられたもの。

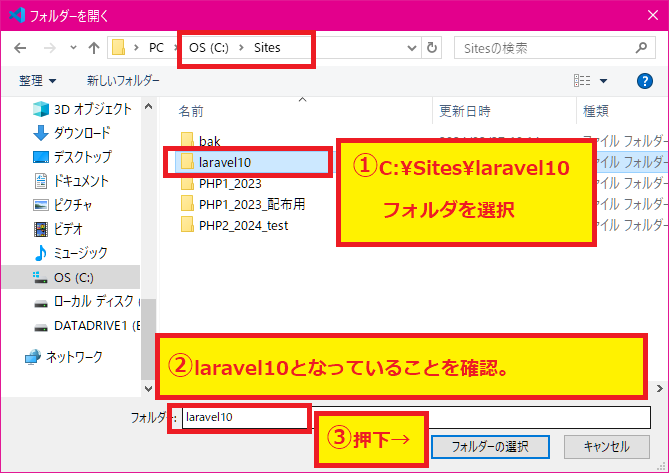


０－７．IDE helperの導入①

1. VS Codeの[ファイル]-[フォルダーを開く]を選択。



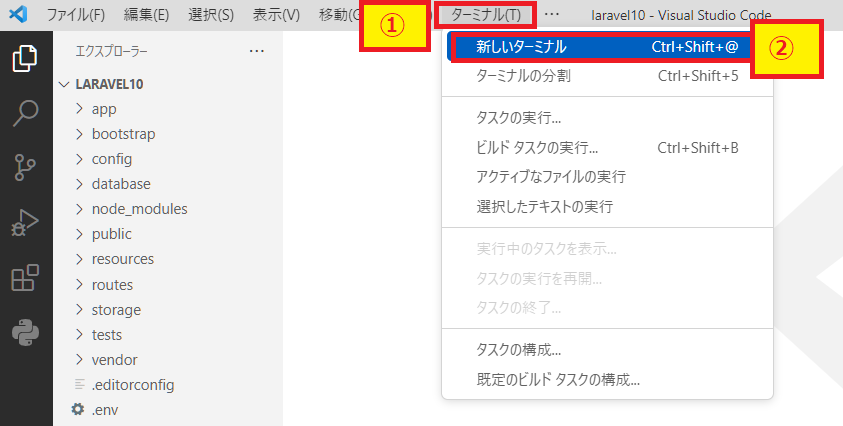
1. [フォルダーを開く]のダイアログで[C:\Sites\laravel10]プロジェクトフォルダを選択。



1. VS Codeで[laravel10]プロジェクトフォルダが開いていることを確認。

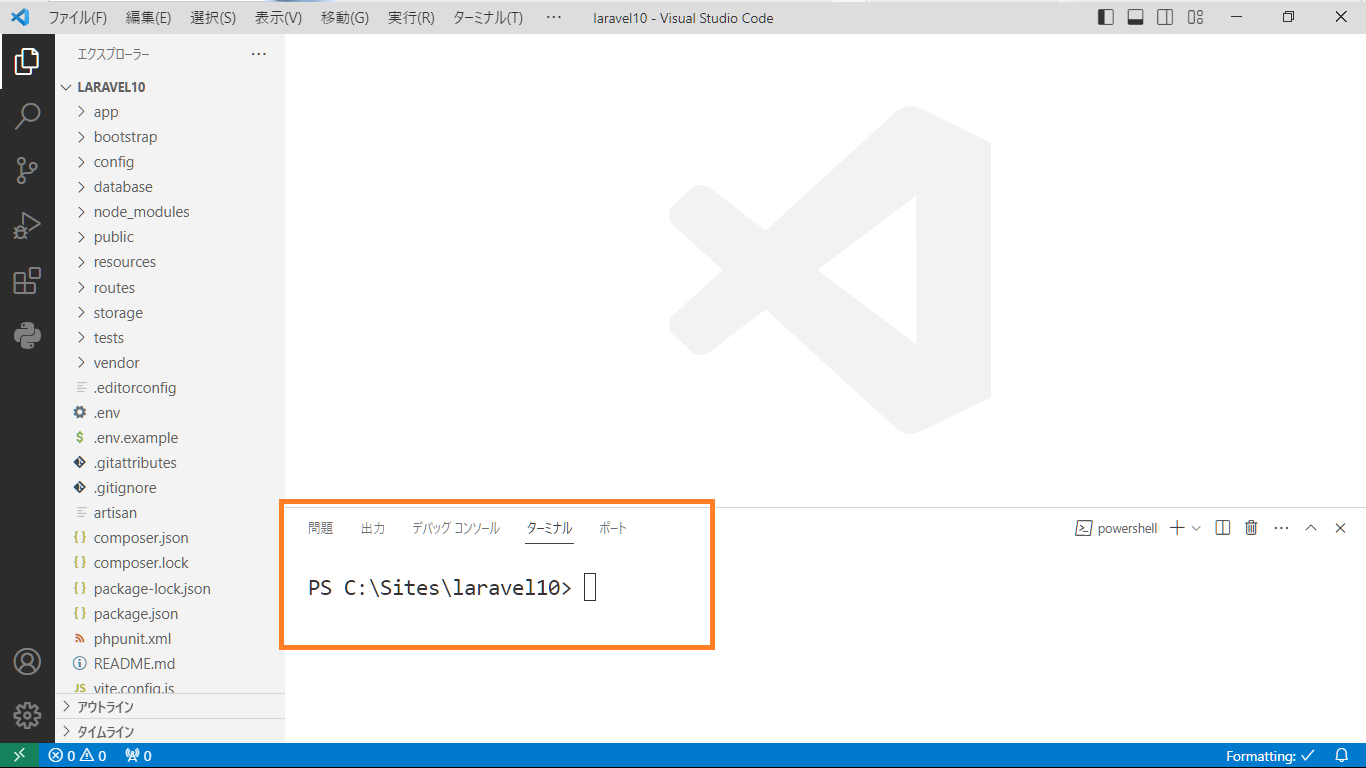


1. VS Code [ターミナル] – [新しいターミナル] を押下。



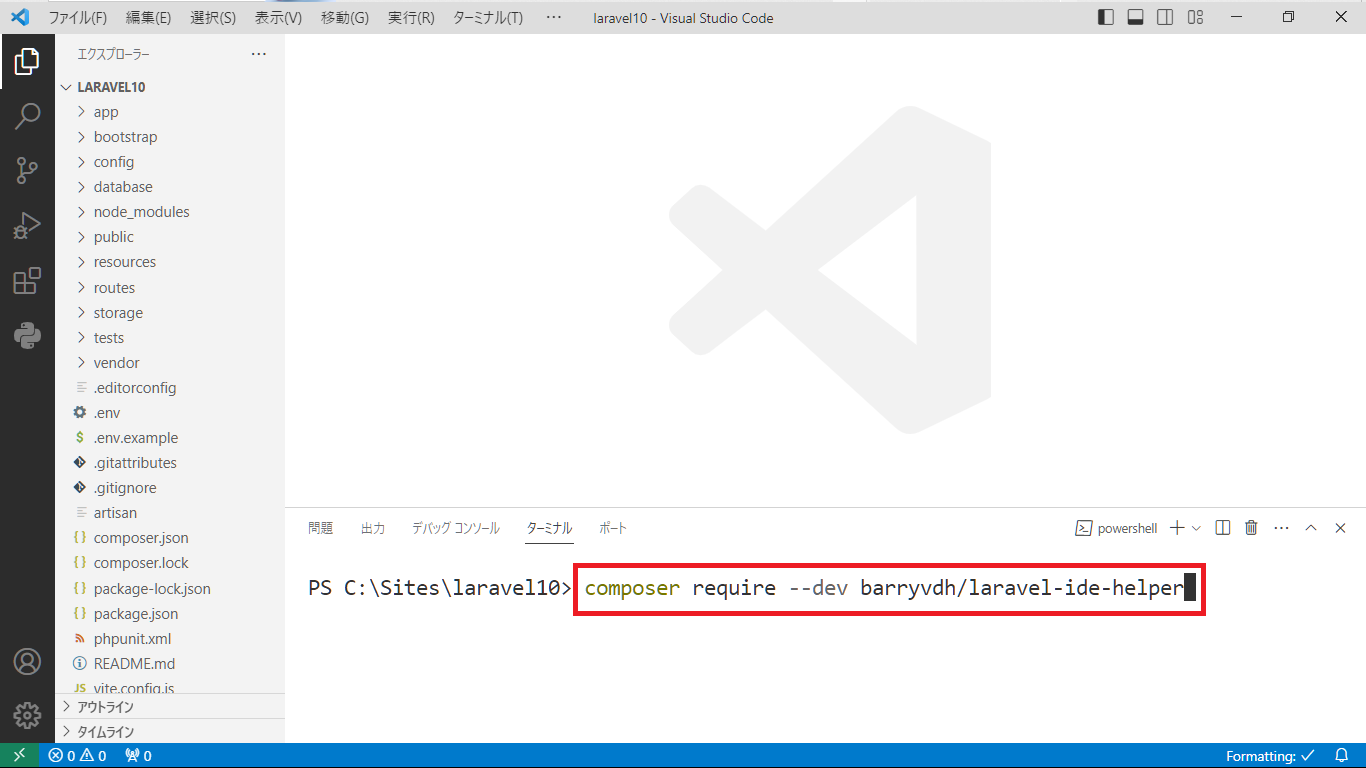
1. VS Code [ターミナル] でlaravelのプロジェクトに入っていることを確認。

※今後、授業内でコマンドを実行する際は、この手順で行います。



1. VS Code [ターミナル] で[composer require --dev barryvdh/laravel-ide-helper]と入力。

※処理が終了したら、ターミナルに入力可能状態となる。



０－８．DB接続の設定とIDE helperの導入②

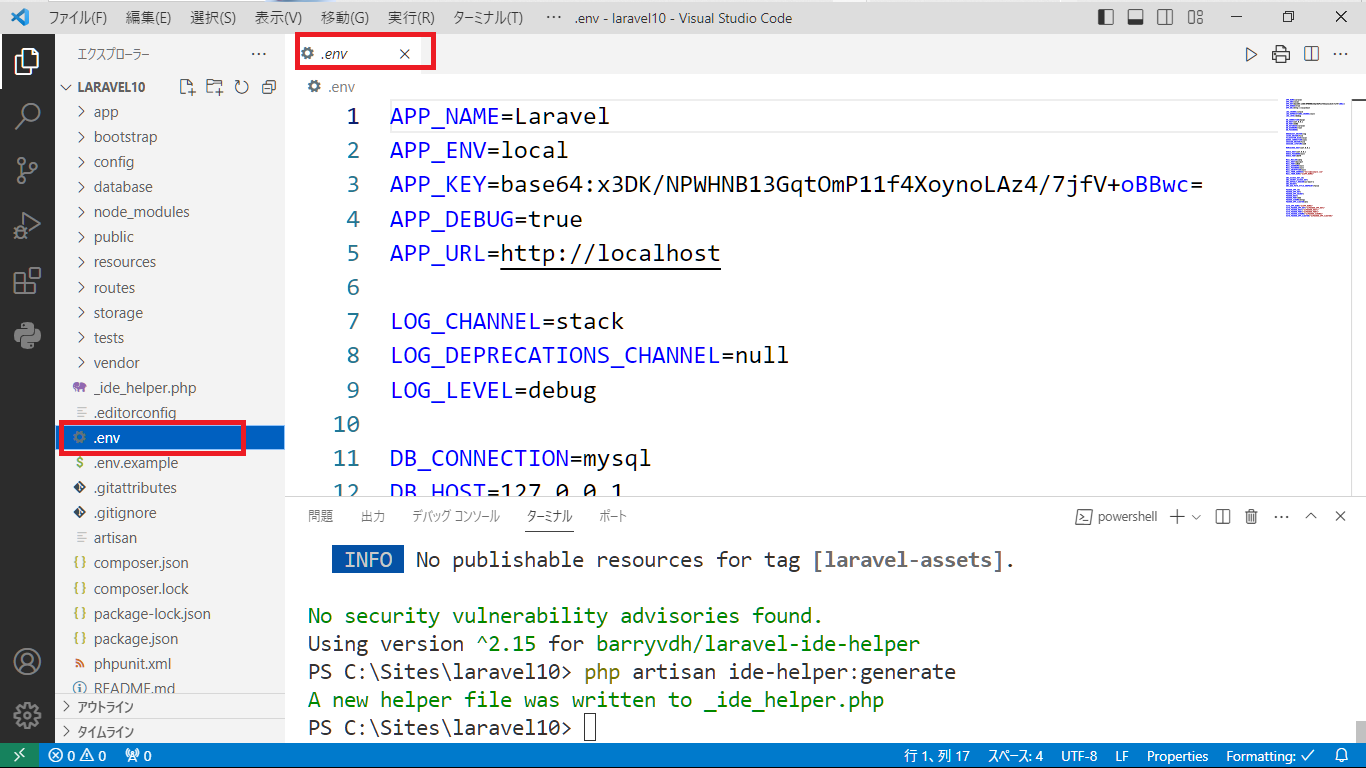
1. VS Code [ターミナル] で[php artisan ide-helper:generate]と入力。

※処理が終了したら、ターミナルに入力可能状態となる。

※[ターミナル]は後ほど使用するので、そのまま開いておく。



1. VS Codeのエクスプローラーで[.env]ファイルを開く。



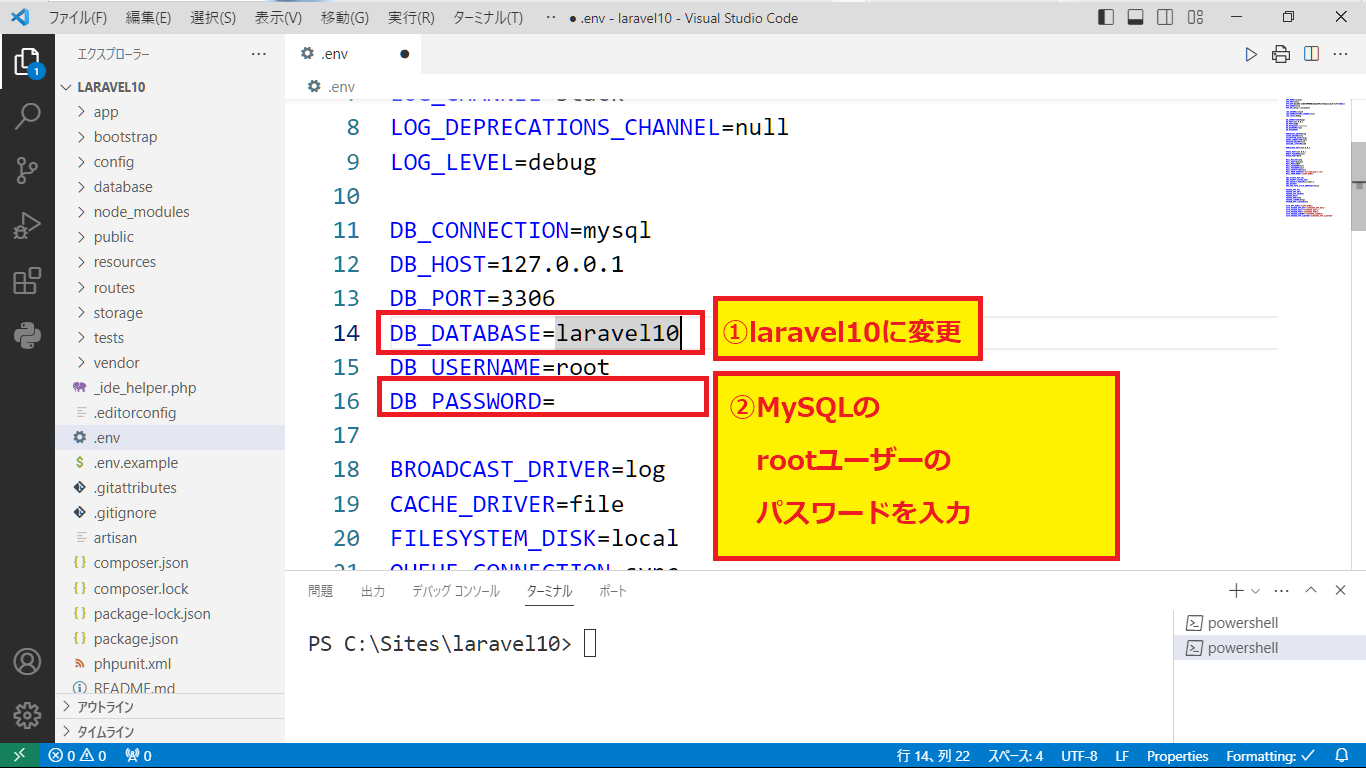
1. [.env]ファイルの以下を編集。

●[DB\_DATABASE=laravel10] に修正。

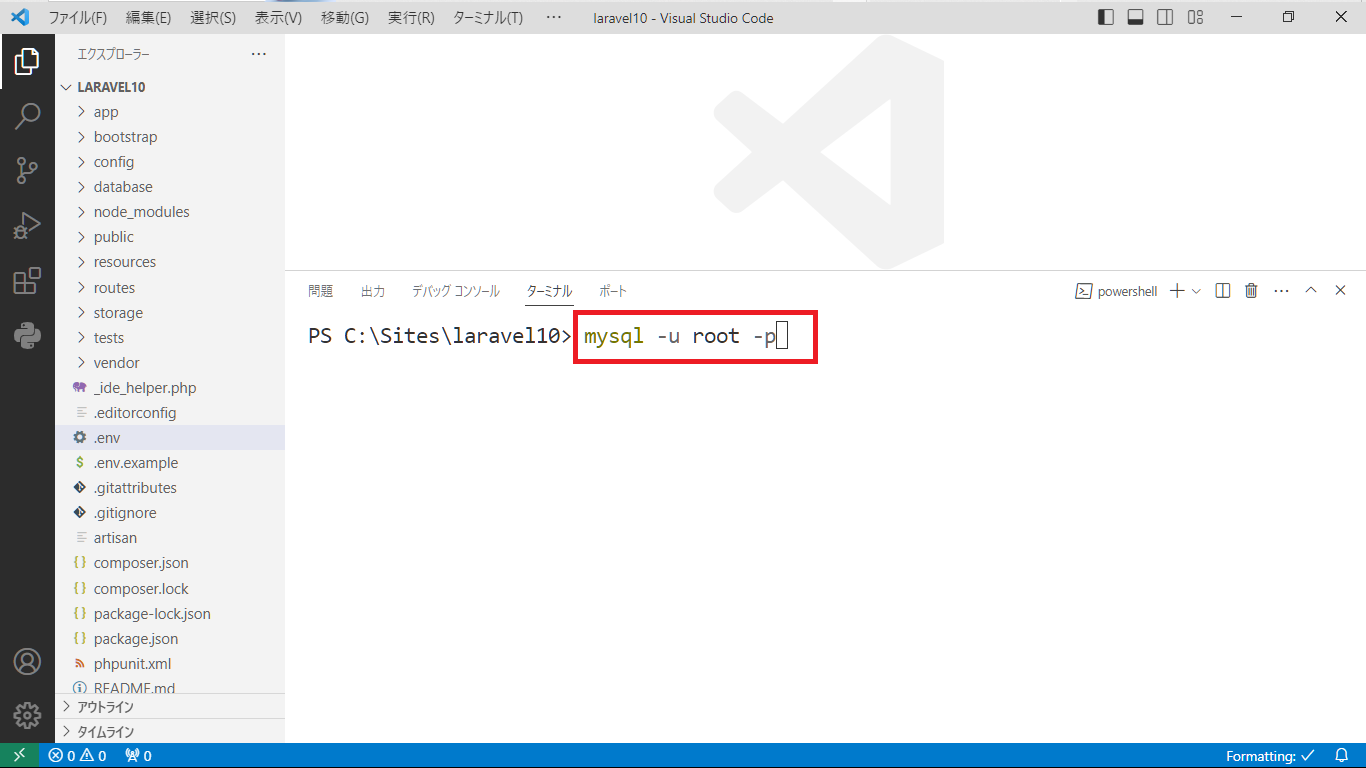
　　→DBはこの後③の手順で作成します。

●[DB\_PASSWORD=]の後に、MySQLのrootユーザーのパスワードを入力。

※上書きしたら、[.env]ファイルは閉じて良い。



1. [ターミナル]でＭｙSQLにrootユーザーでログイン。



1. ＭｙSQLにログインしたら、[laravel10]という名前のDBを作成。

※ターミナルで [CREATE DATABASE laravel10;] を実行。



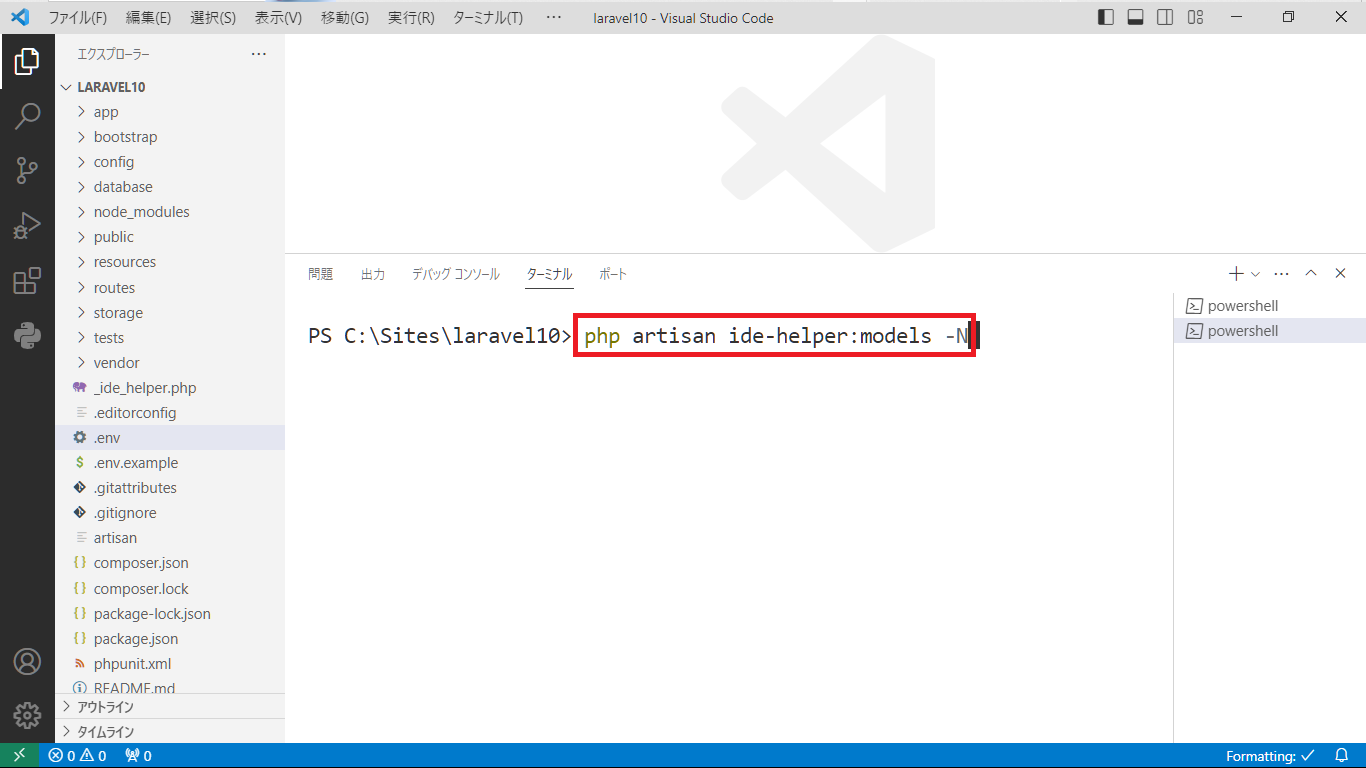
1. ターミナルで [exit;] を実行し、MYSQLからログアウト。



1. ターミナルで [php artisan ide-helper:models -N]を実行。

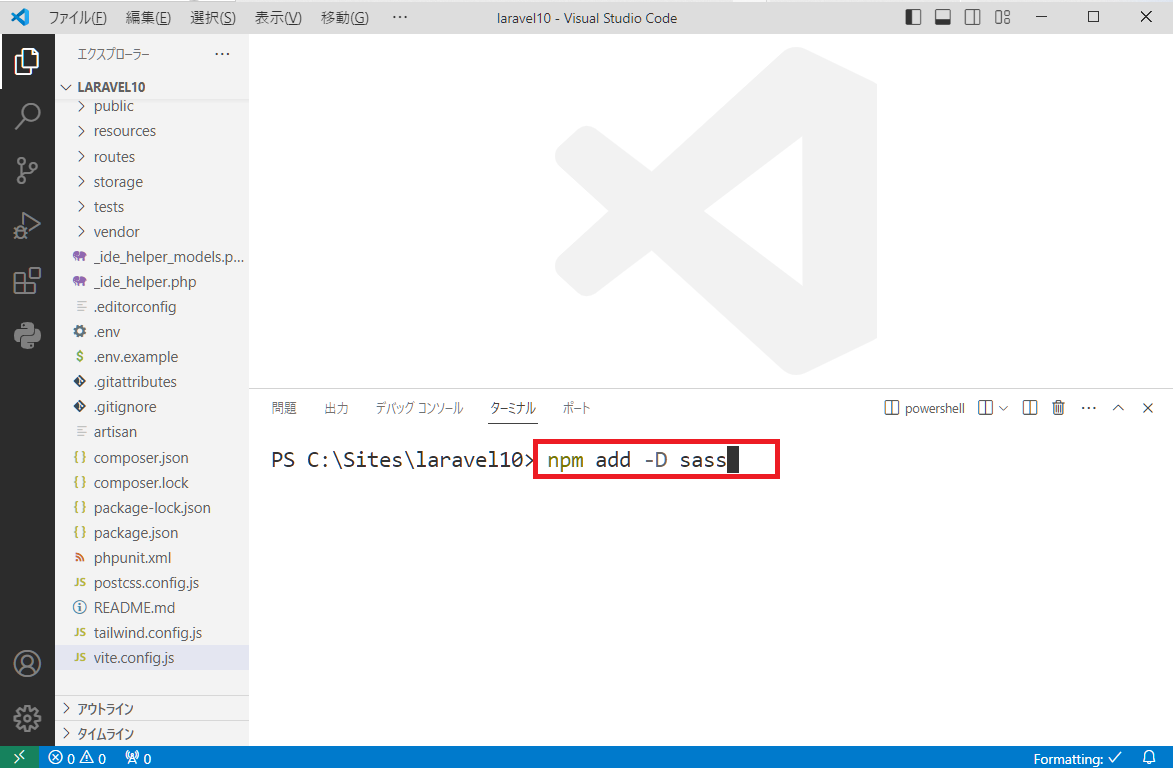
※このコマンドは、DB接続設定が正しくなければ、エラーが発生するので注意。

※[Model information was written to \_ide\_helper\_models.php]と表示されれば完了。

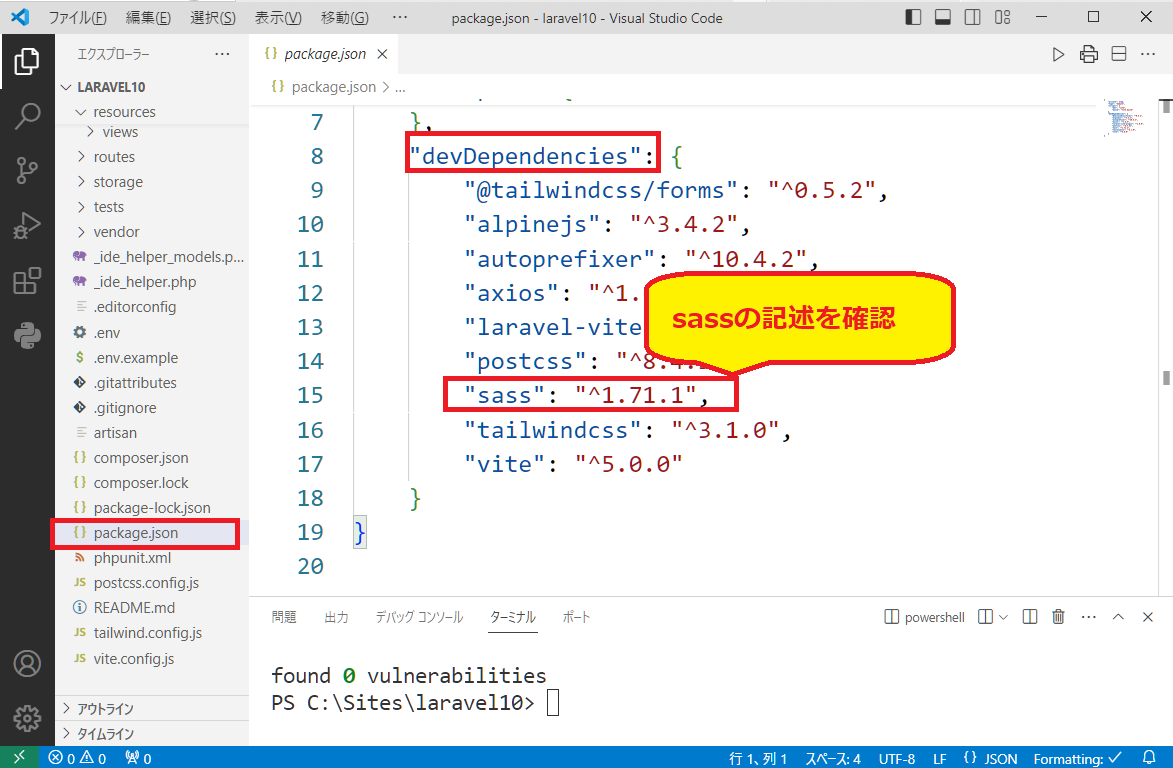


０－９．Tailwindの設定

1. ターミナルで [npm add -D sass]と入力。

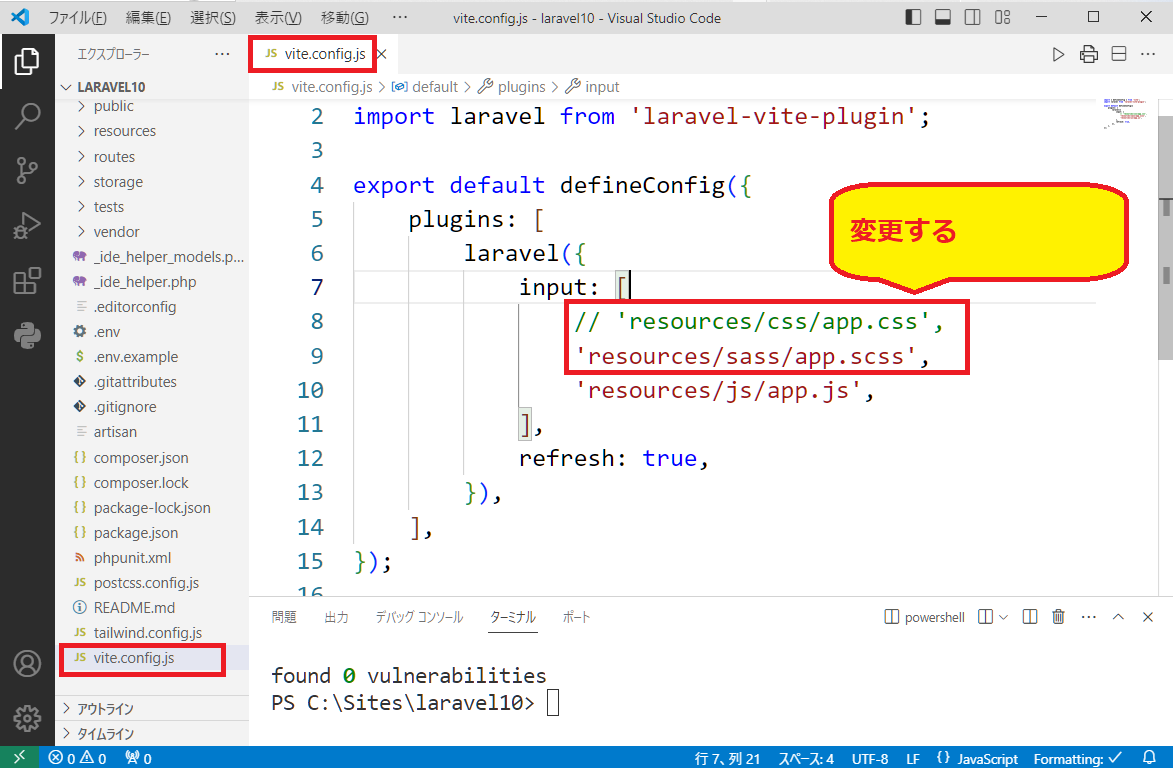


1. [package.json]ファイルを開き、devDependenciesの中に[sass]の記述が存在することを確認。※確認したら、ファイルは閉じてOK。



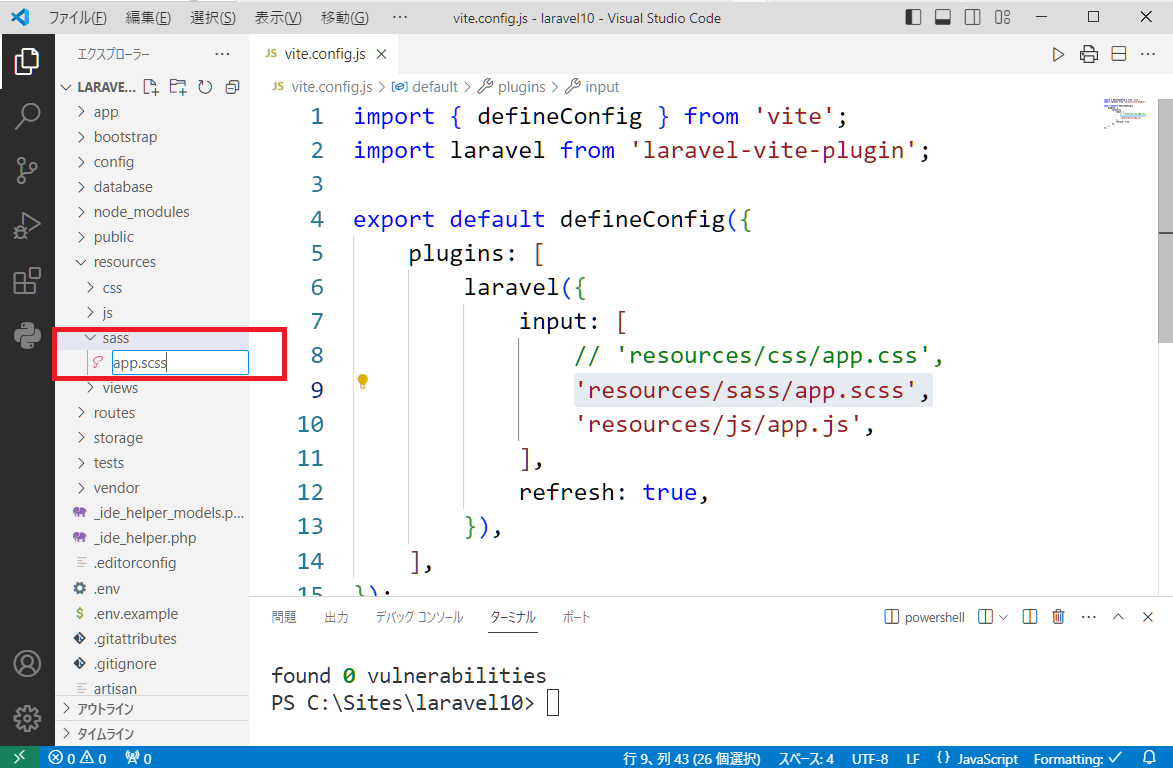
1. [vite.config.js]ファイルを開き、

８行目辺りの、['resources/css/app.css',]　→　['resources/sass/app.scss',]に変更する。※画像の例では、分かり易さのため、元々の記述をコメントアウトにしています。



1. [resources]フォルダの配下に、[sass]フォルダを新規作成し、

[sass]フォルダの配下に、[app.scss]ファイルを新規作成。

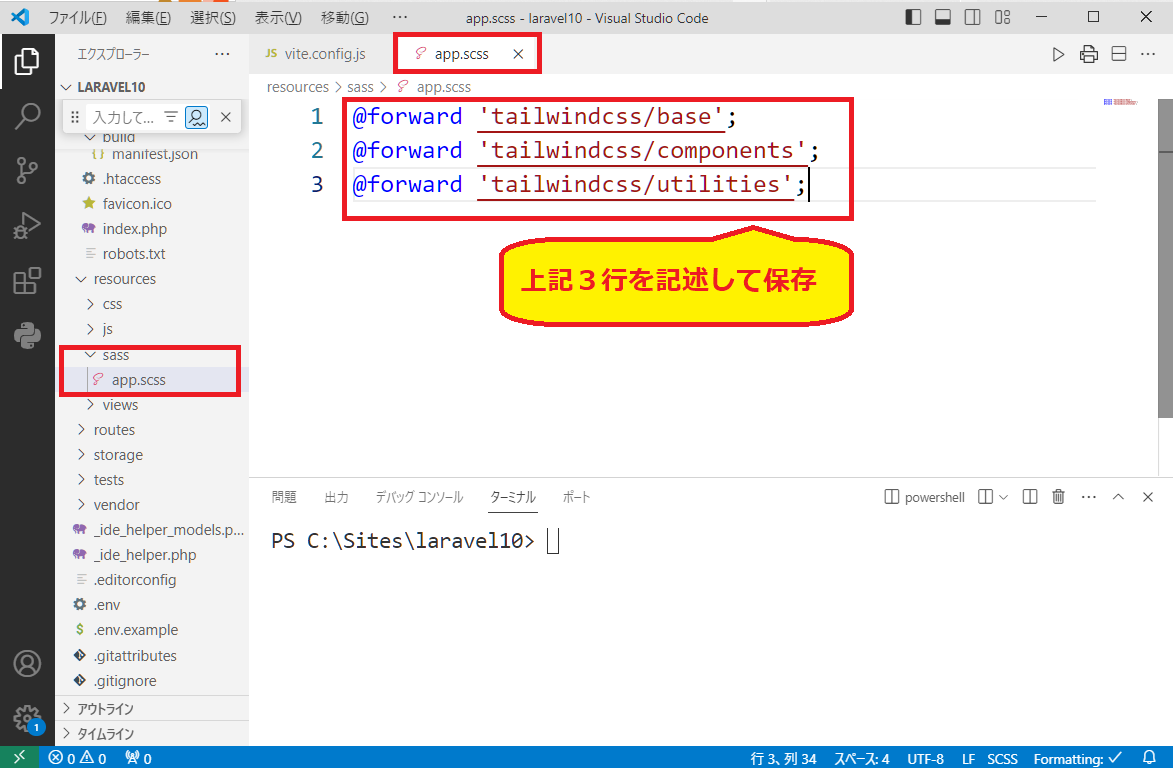


1. [app.scss]ファイルに下記を記載し、上書き保存。

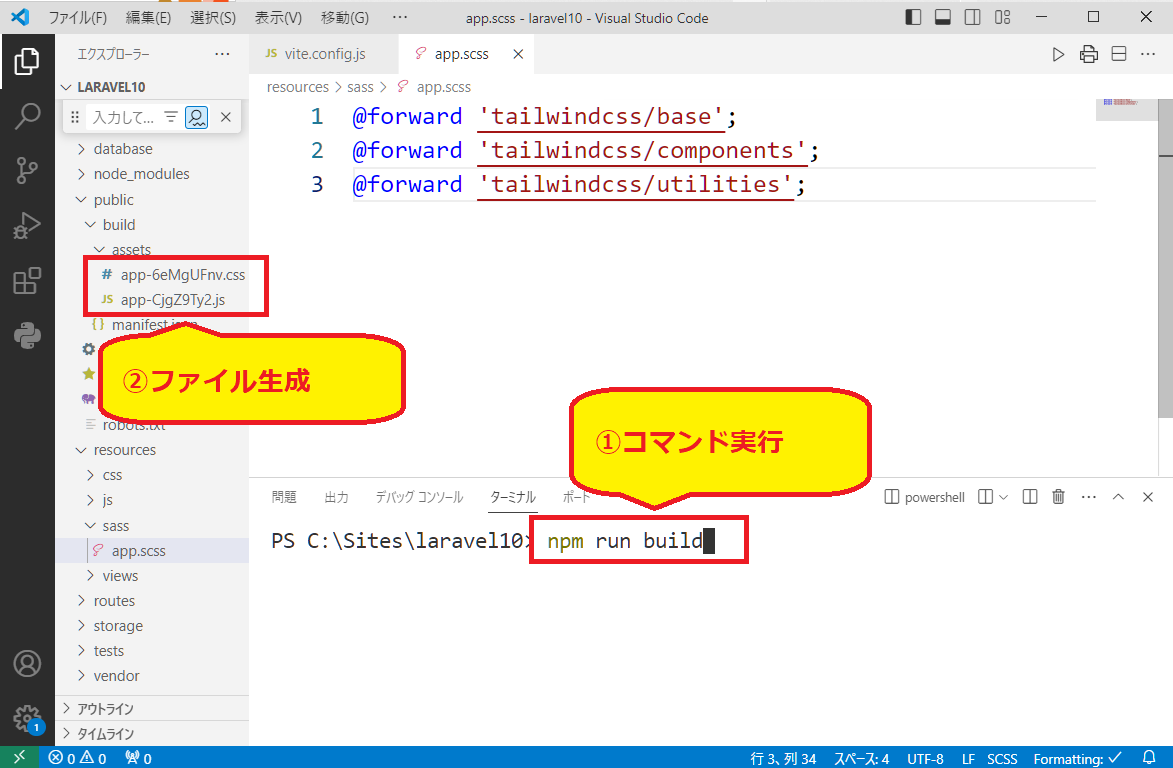
@forward 'tailwindcss/base';

@forward 'tailwindcss/components';

@forward 'tailwindcss/utilities';



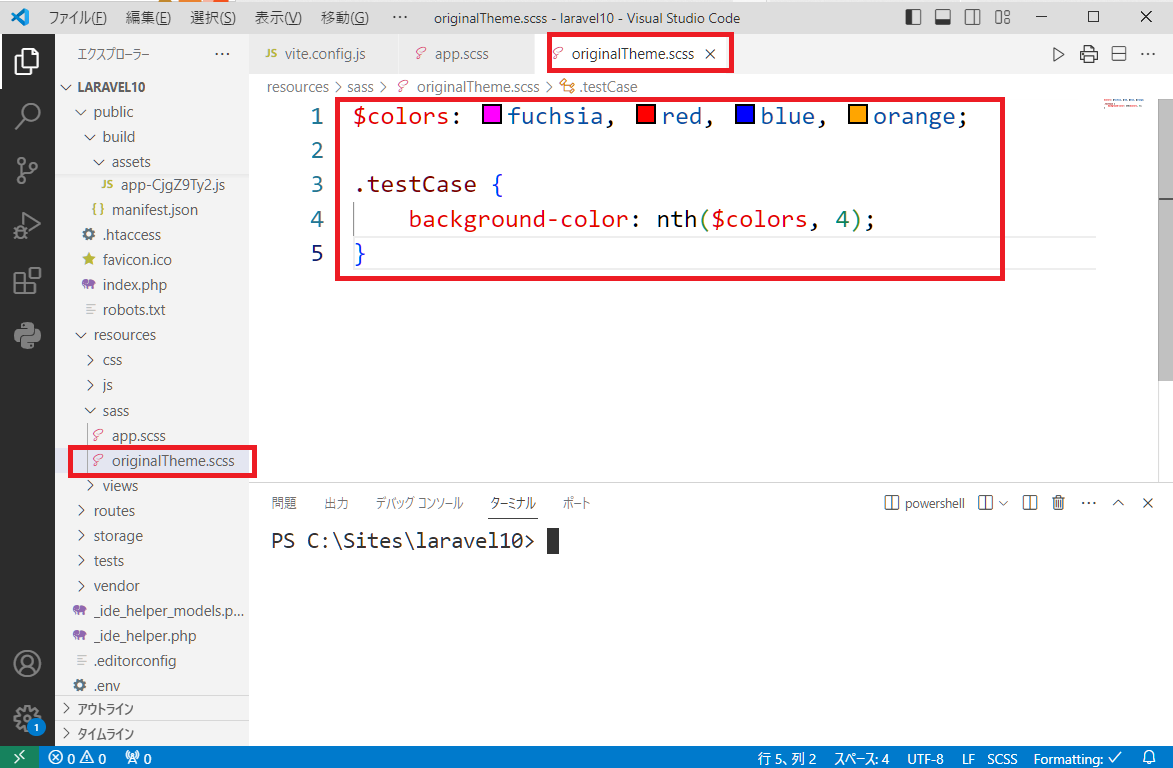
1. ターミナルで [npm run build]を入力。ビルド実行により[public\build\asset]配下にファイルが作成されます。



【自作のcssも使いたい場合】

1. [resources\sass\]配下に拡張子が「.scss」のファイルを作成し、任意のSCSSを記述。

※SCSSの詳細説明については割愛します。

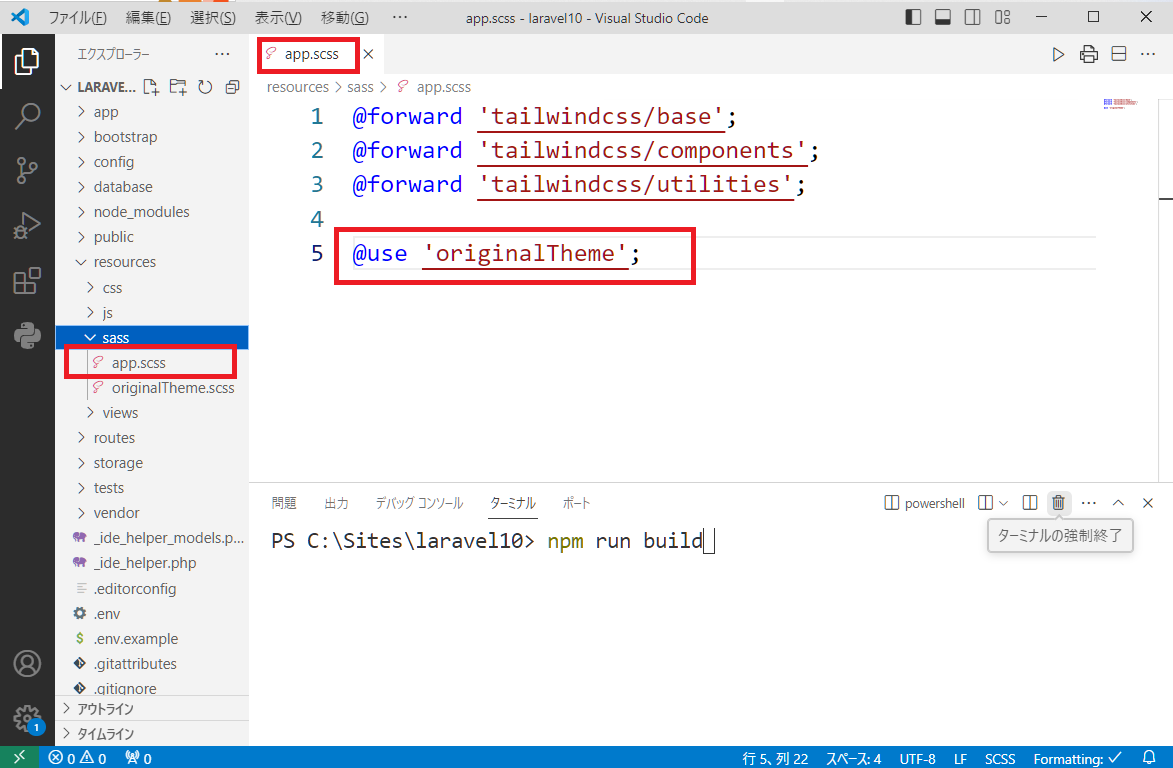


1. [resources\sass\app.scss]ファイル内に、⑦で作成したファイルを呼び出すように記述。

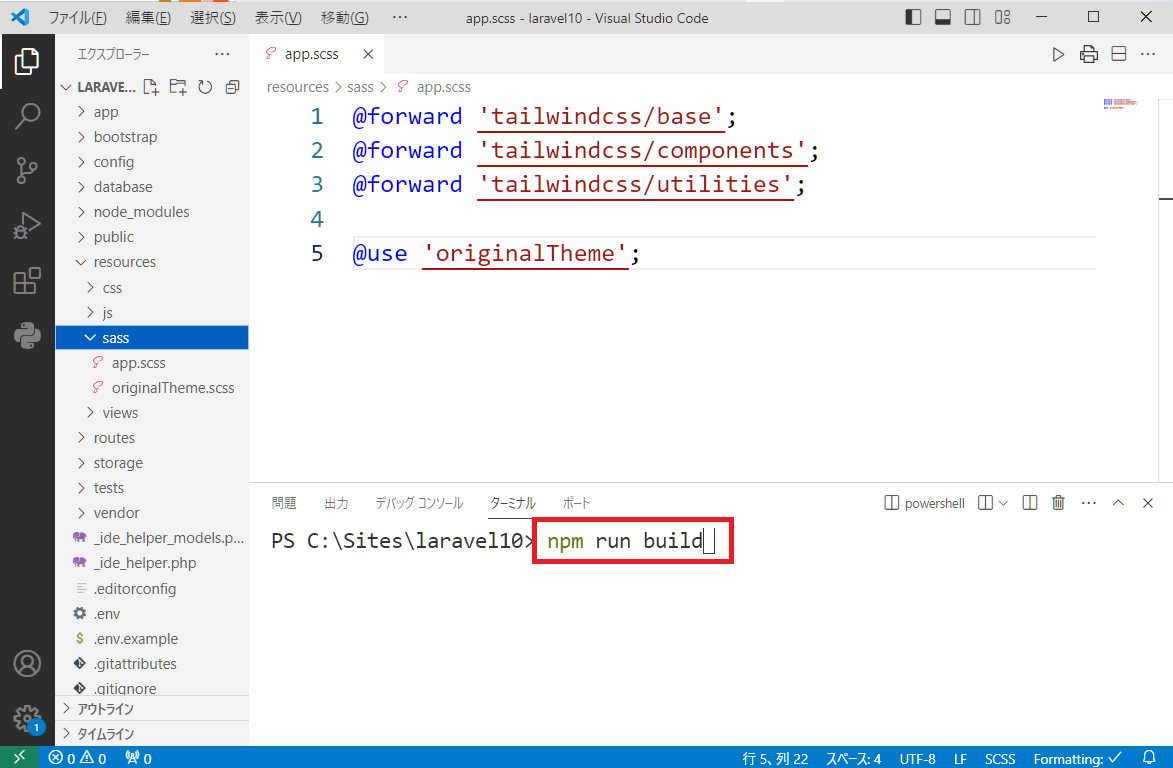
呼び出しには「@forword」と「@use」があり、継承の有無によって使い分けが必要です。

※「@import」もありますが、現在は非推奨です。

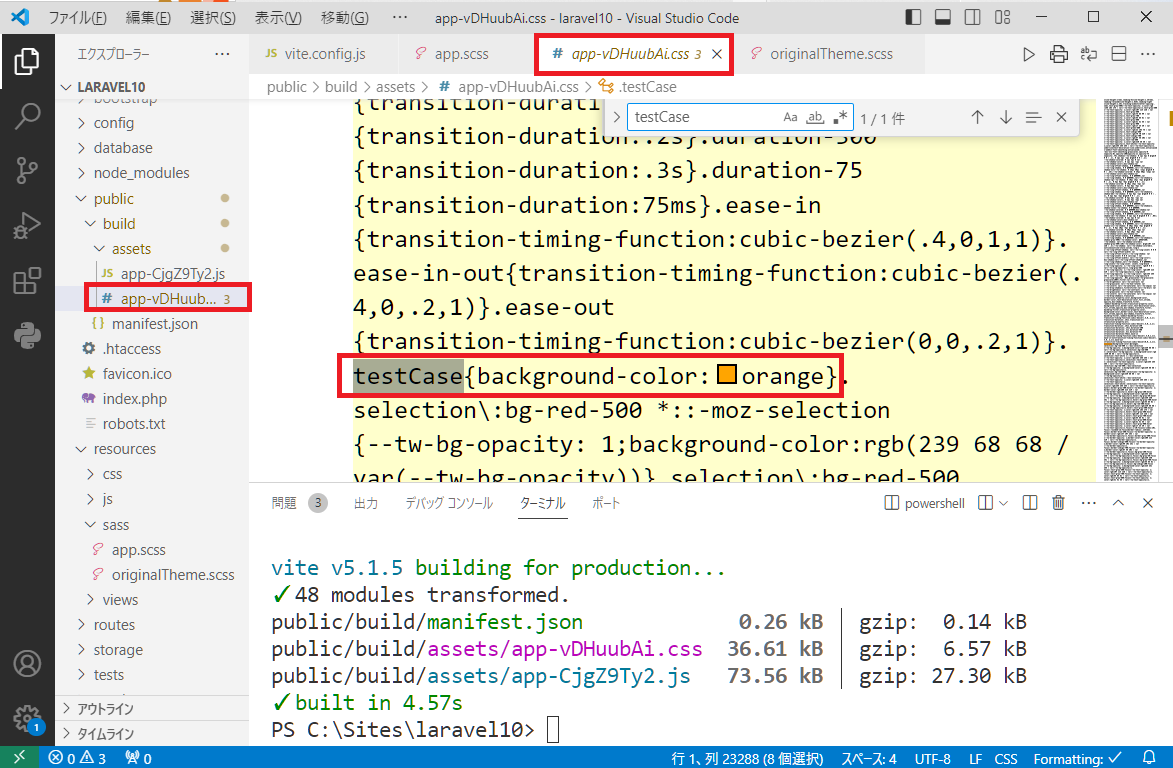
※本授業では、導入部分の紹介のみとします。



1. ターミナルで [npm run build]を入力。ビルド実行により[public\build\asset]配下にファイルが作成されます。



[public\build\assets]下の生成されたファイルに、自作SCSSの内容が書き出されています。



1. ターミナルで [exit]を入力し、ターミナルを終了。

**※以降、ターミナルを終了する際は、必ず[exit]コマンドで終了させること。**

環境構築、および初期設定は以上です。