



3. プロジェクト作成

ショッピングカートの環境

今回は Laravel で簡単なショッピングカートを作成してみましょう。

動作環境

プロジェクトを作成する前に、各ソフトウェアのバージョンは以下を推奨します。

- PHP (7.4.x 以上)
- Composer (2.1.x 以上)
- Node.js (16.x 以上)
- MySQL (5.x 以上)

プロジェクト作成

Composerで作成

ターミナルを開き、 **Composer** コマンドで「shopping」プロジェクトを **Ver.8** で作成します。

Composerで作成する場合

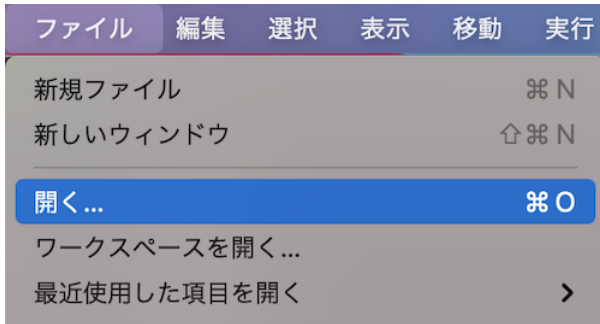
```
% composer create-project laravel/laravel shopping "8.*"
```

Laravelコマンドで作成する場合

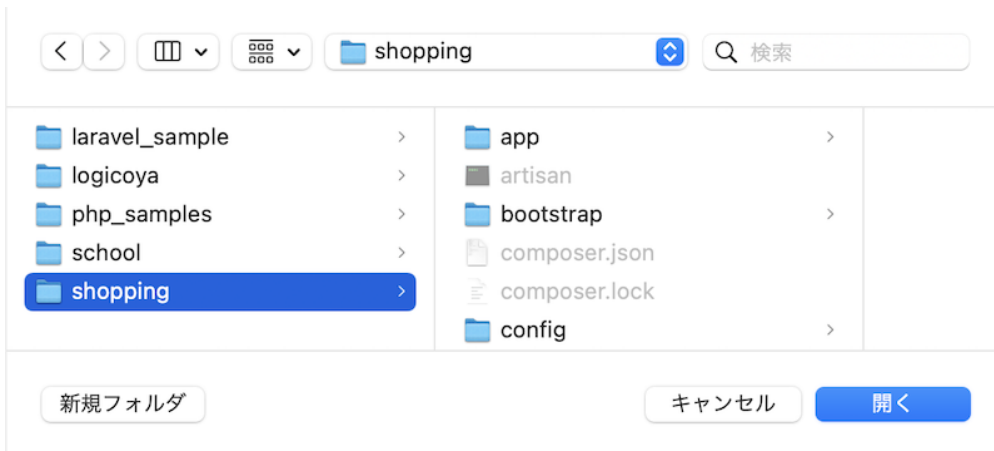
```
% laravel new shopping "8.*"
```

VSCodeでプロジェクトを開く

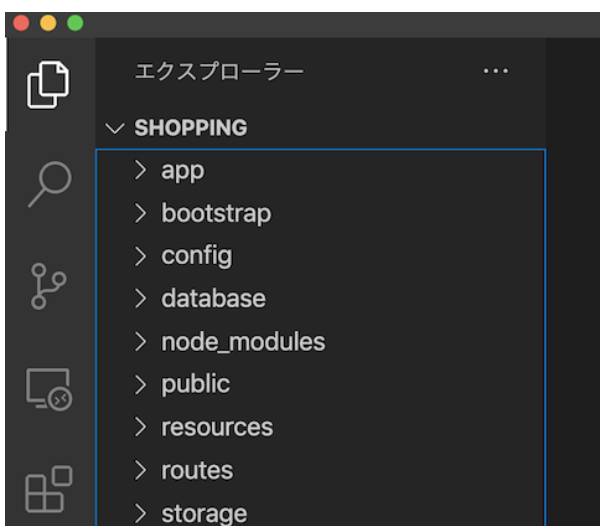
VSCode でフォルダを開きます。



「shopping」 プロジェクトを開きます。



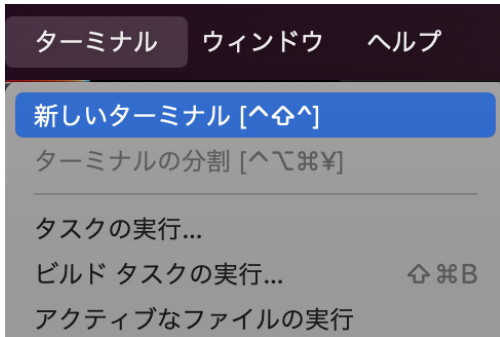
「shopping」 プロジェクトが表示されました。



サーバ起動確認

バージョン確認

VSCode でターミナルを開きます。



artisan コマンドでバージョンを確認します。

```
% php artisan -V
Laravel Framework 8.79.0
```

アクセス権の変更

Mac や Linux の場合は、アクセス権を変更します。Windows の場合は基本的に必要はありません。

```
% chmod -R 777 storage
% chmod -R 777 bootstrap/cache
```

サーバの起動

Laravelサーバを起動します。

```
% php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
```

<http://127.0.0.1:8000> または、 <http://localhost:8000> にアクセスするとアプリが表示されました。



Documentation

Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.



Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.



Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.



Vibrant Ecosystem

Laravel's robust library of first-party tools and libraries, such as [Forge](#), [Vapor](#), [Nova](#), and [Envoyer](#) help you take your projects to the next level. Pair them with powerful open source libraries like [Cashier](#), [Dusk](#), [Echo](#), [Horizon](#), [Sanctum](#), [Telescope](#), and more.

サーバ終了

サーバを修了するには起動中のターミナルで、Windows は **Ctrl + C**、Mac は **Cmd + C** キーを押します。

```
...
[Sun Jan 16 19:49:39 2022] 127.0.0.1:59955 [200]: GET /favicon.ico
[Sun Jan 16 19:49:39 2022] 127.0.0.1:59955 Closing
^C
xxxx shopping %
```

コマンド入力に戻れば、サーバが終了しています。

Laravel Jetstream

Laravel Jetstream とは？

Laravel7 までは `aravel/ui` や `make:auth` コマンドを利用して認証ページを作成できましたが、**Laravel8 の認証機能としてJetstreamが標準搭載**されました。**Jetstream** は、APIサポート (Laravel Sanctum)、ユーザー登録、メール確認、2要素認証、セッション管理など、機能が充実しています。

Jetstream インストール

Jetstream を利用するには、まず **Composer** でインストールします。

```
% composer require laravel/jetstream
```

Livewire または Inertia の選択

Jetstream は、CSSフレームワーク「Tailwind CSS」を標準搭載しており、**Livewire** または **Inertia**（イナーシャ）のどちらかでレイアウト選択します。また **Laravel 7** 以前のように **laravel/ui** で開発することも可能ですが、将来的には **Livewire** または **Inertia** に対応しておいた方が良いでしょう。

Livewire

Laravelの標準テンプレートエンジンは **Blade** ですが、**Bladeをよりダイナミックに活用できるのがLivewire**です。フロントエンドはJSフレームワークである **React** や **VueJS** を中心に開発していましたが、**Livewire** を利用すると**JSでコーディングする必要がありません。**

LivewireによるPHPの例

```
use Livewire\Component;

class SearchUsers extends Component
{
    public $search = '';

    public function render()
    {
        return view('livewire.search-users', [
            'users' => User::where('username', $this->search)->get(),
        ]);
    }
}
```

LivewireによるBladeの例

```
<div>
    <input wire:model="search" type="text" placeholder="Search users..." />
</div>
```

```
<ul>
  @foreach($users as $user)
    <li>{{ $user->username }}</li>
  @endforeach
</ul>
</div>
```

Inatial

React や **VueJS** などの **JSでフロントエンド開発するには Inatiaを選択**します。 **Livewire** は **Laravel**の独自機能のため、学習コストが高くなるのがデメリットです。逆に **JS** の方が圧倒的に開発人数が多いため、今回は **Inatial** を採用します。

Inatia のインストール

artisan コマンドで、 **Inatial** をインストールします。

```
% php artisan jetstream:install inertia
```

インストールが完了すると、 **npm** インストール&ビルドするよう促されます。

```
...
Please execute "npm install && npm run dev" to build your assets.
```

npmインストール

npm でパッケージをインストールします。

```
% npm i
```

しばらくしてインストールが完了したら、開発用にビルドします。

```
% npm run dev
```

ビルドが成功しました。

```
✓ Compiled Successfully in 1491ms
...
webpack compiled successfully
```

データベース作成

データベースの作成

MySQLコマンドやDB管理ツールで、データベースを作成してみましょう。

MySQLコマンドで作成する場合

ターミナルでMySQLにログインします。ユーザ名、パスワードはMySQLの設定時のものを利用してください。

```
% mysql -u root -p
```

MySQLにログインしたら、「laravel_shopping」でデータベースを作成します。

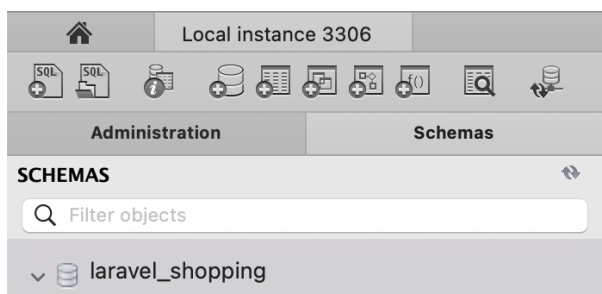
```
mysql> create database laravel_shopping;
```

データベース確認

データベースを作成したらログアウトします。

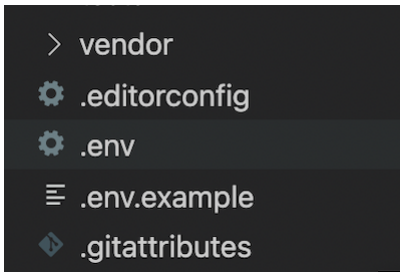
```
mysql> exit
```

DBツールで「laravel_shopping」データベースを確認してみましょう。



データベース設定

Laravel でデータベースを利用するには、まず、`.env` ファイルにその情報を記述します。



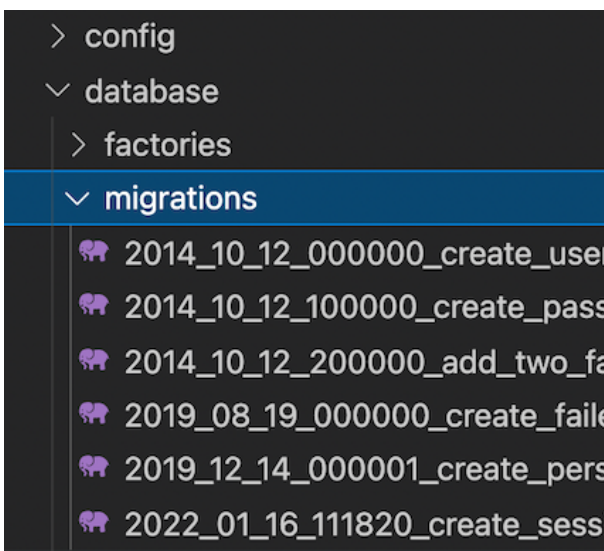
`.env` ファイルの `DB_DATABASE` の値を「`laravel_shopping`」に変更します。その他、ユーザ名とパスワードはMySQLの設定にあわせてください。

`.env`

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_shopping
DB_USERNAME=root
DB_PASSWORD=
```

マイグレート(migrate)

プログラムでDBを管理することをマイグレーション(migration)といいます。`Laravel` プロジェクト作成すると、`database/migrations/` に、いくつかのDBの定義プログラムが用意されています。



Laravelでマイグレート

artisan コマンドでマイグレートしてみましょう。

```
% php artisan migrate
```

マイグレートが成功しました。

```
Migration table created successfully.  
Migrating: 2014_10_12_000000_create_users_table  
Migrated: 2014_10_12_000000_create_users_table (15.53ms)  
Migrating: 2014_10_12_100000_create_password_resets_table  
Migrated: 2014_10_12_100000_create_password_resets_table (6.35ms)  
Migrating: 2014_10_12_200000_add_two_factor_columns_to_users_table  
Migrated: 2014_10_12_200000_add_two_factor_columns_to_users_table (5.88ms)  
Migrating: 2019_08_19_000000_create_failed_jobs_table  
Migrated: 2019_08_19_000000_create_failed_jobs_table (6.56ms)
```

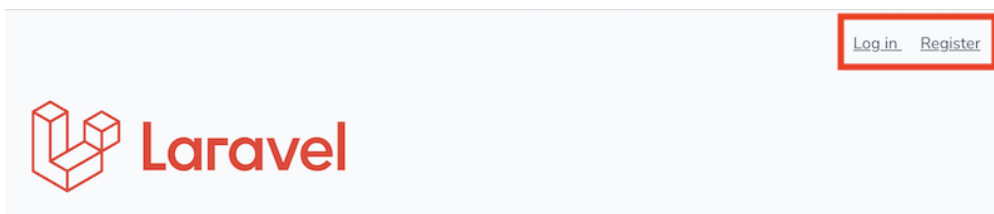
ユーザ認証確認

リンクの確認

Laravelサーバを起動します。

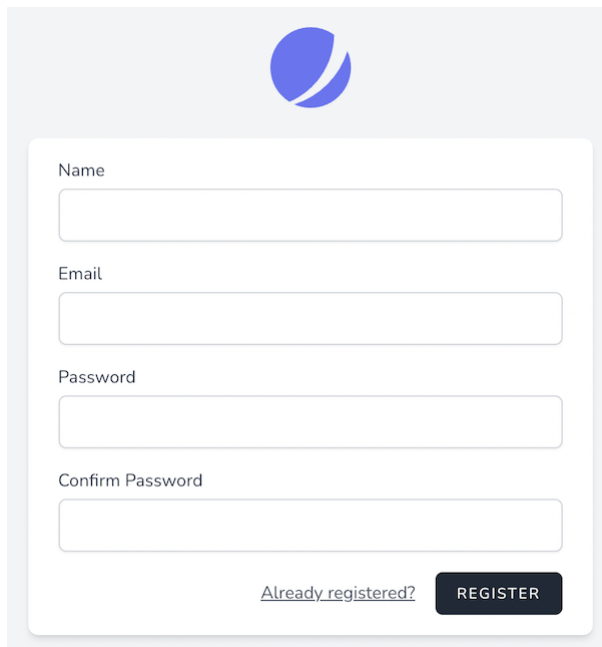
```
% php artisan serve
```

ブラウザでトップページを確認すると「Log in」「Register」のリンクが自動的に作成されています。



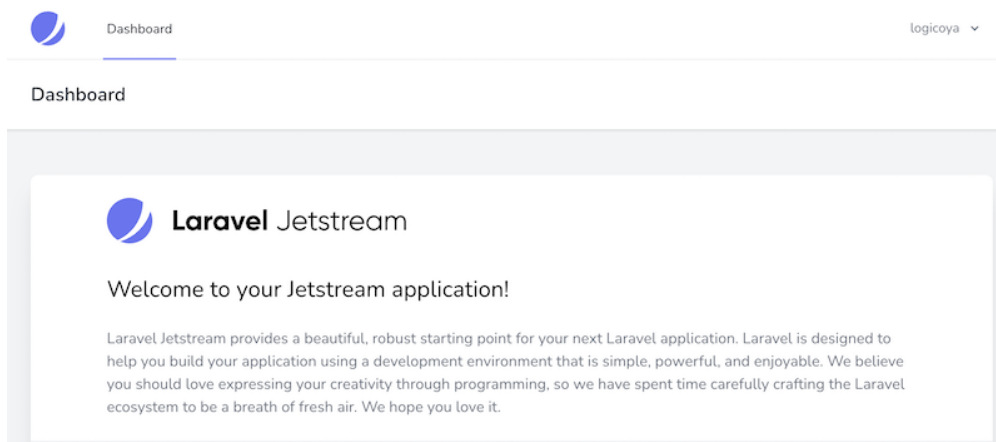
会員登録

「register」にアクセスすると、入力画面が表示されるので会員登録をしてみましょう。



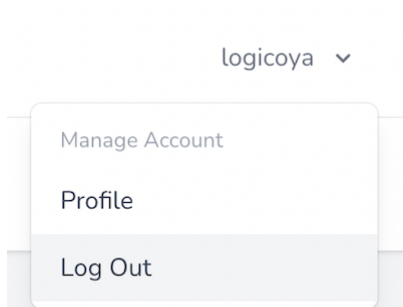
Registration form for Laravel Jetstream. It includes a blue circular logo at the top. The form has four input fields: Name, Email, Password, and Confirm Password. Below the Confirm Password field, there is a link [Already registered?](#) and a dark blue button labeled **REGISTER**.

登録完了すると、ユーザページにログインします。



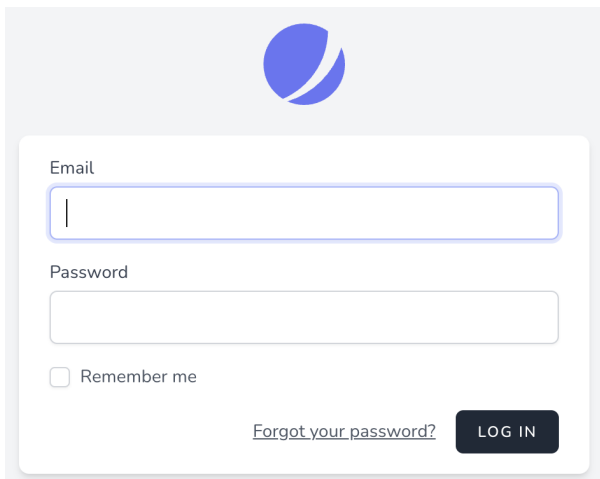
ログアウト

ログアウト機能も作成されています。ユーザ名をクリックして「Log Out」でログアウトします。



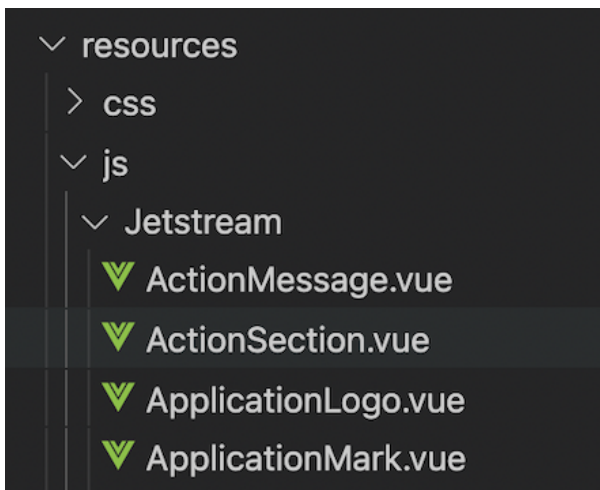
ログイン

ログイン画面から、ログインできるか確認してみましょう。



デフォルトは VueJS

`resources/js/` に、`.js` や `.vue` ファイルなどが入っているので確認してみましょう。



Laravel で Inertia を選択すると、会員登録やログイン機能はデフォルトでVueJSが採用されています。 React にすることも可能です。

認証機能はカスタマイズが必要

このように、面倒な認証機能はコマンドだけで作成することができます。ただ、本運用するには会員情報やマルチログインなど、さまざまな機能が必要でしょう。ただ、現段階での実装方法は少々難しいため、別の章で説明します。

当サイトの教材をはじめとするコンテンツ（テキスト、画像等）の無断転載・無断使用を固く禁じます。これらのコンテンツについて権利者の許可なく複製、転用等する事は法律で禁止されています。尚、当ウェブサイトの内容をWeb、雑誌、書籍等へ転載、掲載する場合は「ロジコヤ」までご連絡ください。