

概要

Breeze · Component

講座の紹介



マルチログインに対応したECサイト

前半

section1～3 事前準備

section 4 マルチログイン対応

後半

section 5～7 管理者・オーナー・ユーザー

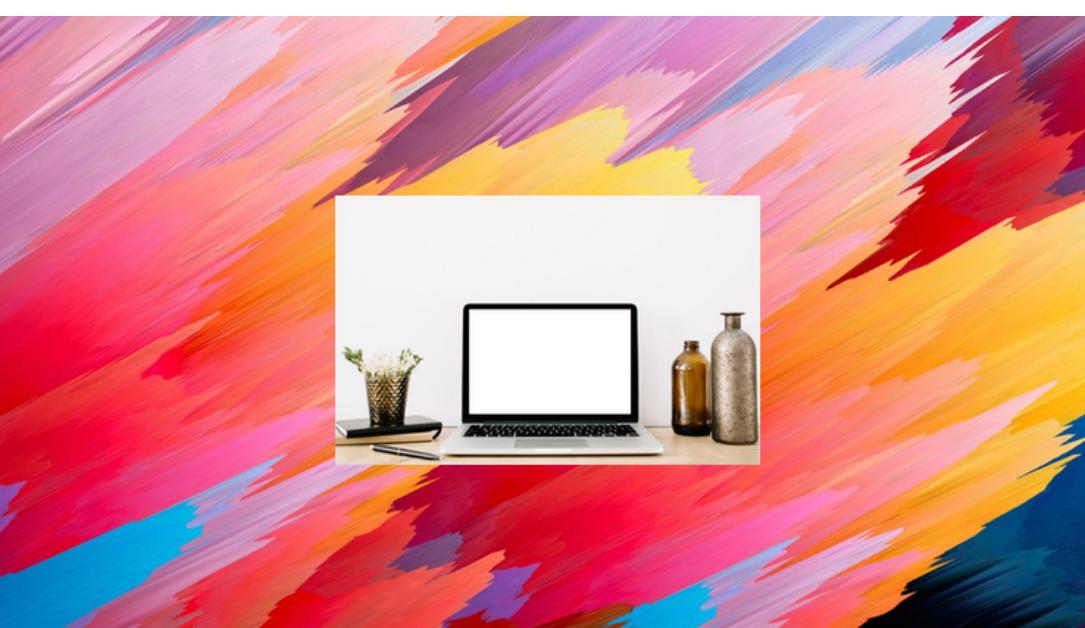
追って追加収録予定

講座の内容

講座の内容

物販サイト/マルチログイン/決済

Laravel6～以降の機能
前回解説していなかった事を多めに

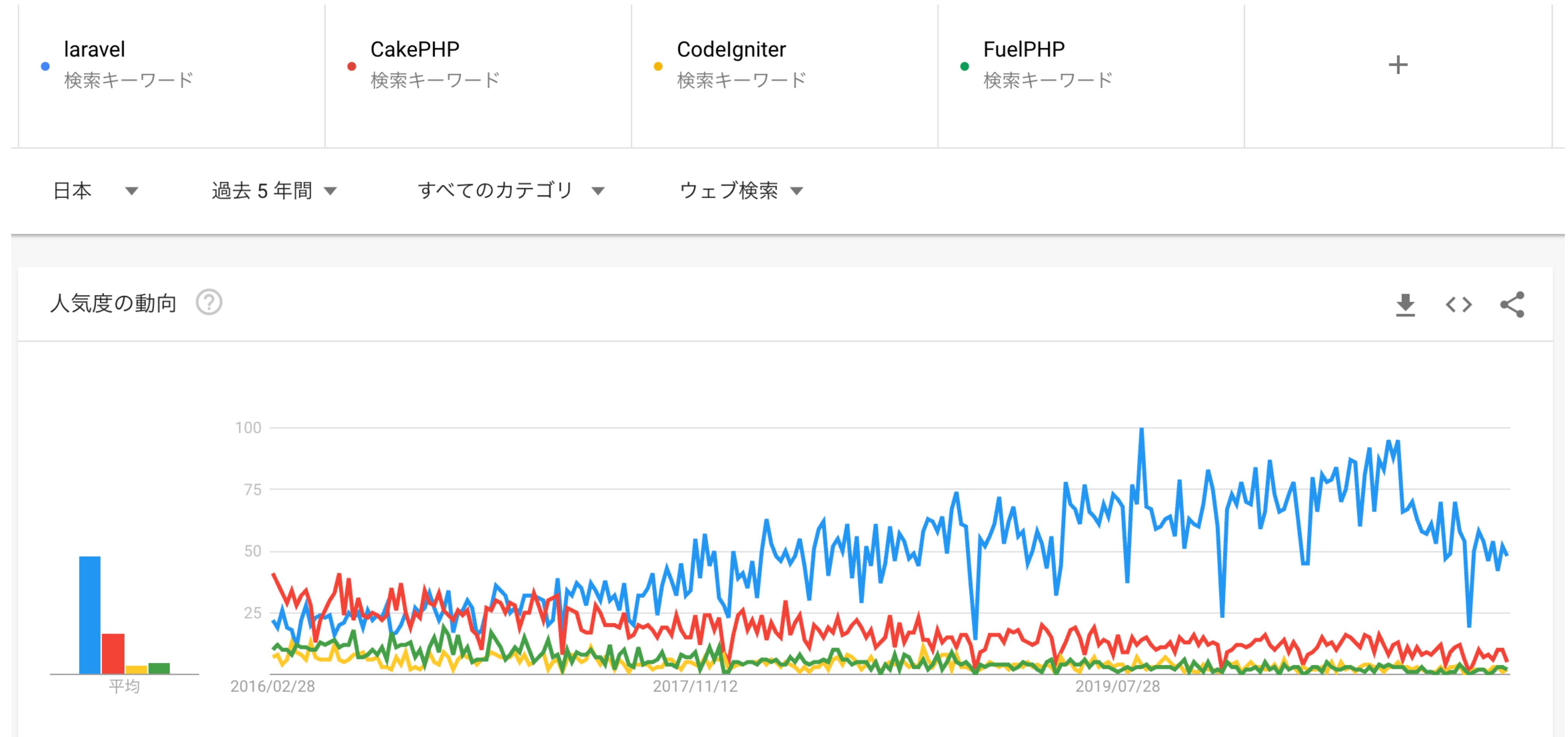


PHPからLaravelまで

マルチログインの応用例

サイトの種類	提供側(販売側)	利用側(購入側)
物販	商品の登録	商品を探す・買う
不動産	物件の登録	物件を探す
求人	求人情報の登録	求人情報を探す
副業	スキルの登録	依頼する
家電修理	エアコンなどの修理内容を登録	探す・依頼する

PHPフレームワークのトレンド



Laravelの年表



2011年 Laravel 1.0

2017年1月 Laravel5.4 PHP5.6.4以上

2017年8月 LTS Laravel5.5 PHP7以上

2019年9月 LTS Laravel6.0 PHP7.2以上

2020年2月 Laravel7.0 PHP7.2以上

2020年9月 Laravel8.0 PHP7.3以上

2021年9月(予定) Laravel9.0 次期LTS

1年に1回バージョンアップ

主な変更内容



Laravel6.0 LTS Laravel UI

Laravel7.0 Sanctum, Bladeコンポーネント

Laravel8.0 Jetstream+livewire/inertia
tailwindcss採用
model/factory/routing記述変更 など

PHPの年表



1995 PHP1.0

2004 PHP5.0.0

2014 PHP5.6.0

2015 PHP7.0

2018 PHP7.3 (Laravel 8～)

2019 PHP7.4

2020 PHP8.0

1年に1回バージョンアップ

設計資料

基本設計リンク

URL設計、テーブル設計、機能設計
(Googleスプレッドシート)

[https://docs.google.com/spreadsheets/d/
1YIDqTKH2v2-
n97kb2GNhWrcMGnJD84JMcTuzD_poMqo/
edit?usp=sharing](https://docs.google.com/spreadsheets/d/1YIDqTKH2v2-n97kb2GNhWrcMGnJD84JMcTuzD_poMqo/edit?usp=sharing)

ER図(draw.io)

[https://drive.google.com/file/d/18sEk5LC-jJum-
NU9JKNZibGRVX81aWE1/view?usp=sharing](https://drive.google.com/file/d/18sEk5LC-jJum-NU9JKNZibGRVX81aWE1/view?usp=sharing)

GitHub



https://github.com/aokitashipro/laravel_umarche

git clone https://github.com/aokitashipro/laravel_umarche

git clone -b ブランチ名 https://github.com/aokitashipro/laravel_umarche



インストール と初期設定

この講座で扱う環境



XAMPP/MAMP

->PHP7.3以上 + サーバー(apache) +
mysql(mariaDB)

Composer->phpライブラリ管理ソフト

Git->バージョン管理ソフト

GoogleChrome / Visual Studio Code

Laravelインストール&起動



```
composer create-project laravel/laravel  
umarche "8.*" --prefer-dist
```

```
cd umarche  
php artisan serve
```

初期設定



Mysql(MariaDB) DB作成

.env 設定(環境ファイル)

config/app.php タイムゾーン、言語設定

デバッグバーのインストール

```
composer require barryvdh/laravel-debugbar
```

もしDB接続できなかつたら



原因は様々あるようですので、
ブログを参照ください。

[https://coinbaby8.com/
access_denied.html](https://coinbaby8.com/access_denied.html)

gitHubでリポジトリ作成



この講座では
gitHubでブランチを分けながら
開発を進めていきます。

設定は必須ではありませんが
もしgit/gitHubも扱いたい場合は
補足1の動画や資料も参考にしてみてください。

タイムゾーン、言語設定



タイムゾーン / 言語設定

config/app.php

‘timezone’ => ‘Asia/Tokyo’;

‘locale’ => ‘ja’;

デバッグバー



デバッグバーのインストール

composer require barryvdh/laravel-debugbar

php artisan serve で確認

.env の DEBUG で 表示切り替えできる

.envファイル更新反映しないなら



php artisan cache:clear

php artisan config:clear

これらのコマンドでキャッシュを消して
再度試してみてください。



Laravel Breeze

認証ライブラリ比較

	Laravel / ui	Laravel Breeze	Fortify	Jetstream
Version	6.x～	8.x～	8.x～	8.x～
View (PHP)	Blade	Blade	-	Livewire + Blade
JS	Vue.js / React.js	Alpine.js	-	Inertia.js + Vue.js
CSS	Bootstrap	Tailwindcss	-	Tailwindcss
追加ファイル	View/Controller/Route	View/Controller/Route	-	View/Controller/Route
機能1	ログイン、ユーザー登録、パスワードのリセット、 メール検証、パスワード確認			
機能2	-	-	2要素認証、 プロフィール管理、チーム 管理	APIサポート (Sanctum)

Laravel Breeze インストール



```
composer require laravel/breeze "1.*" --  
dev
```

```
php artisan breeze:install  
npm install && npm run dev
```

マニュアル: スターターキット

もし表示がおかしかったら

package.jsonにはlaravel-mix指定だけど
layouts/guest.blade.phpやlayouts/app.blade.phpで
@vite が使われているとうまく表示されなくなります。

下記URLより cssとjsの読み込みをお願いします。

https://github.com/aokitashipro/laravel_umarche/blob/main/resources/views/layouts/app.blade.php

```
// コメントアウト
<!-- @vite([]) -->
// 追記
<link rel="stylesheet" href="{{ asset('css/app.css')}}">
<script src="{{ asset('js/app.js')}}" defer></script>
```

Laravel Breeze 追加ファイル(抜粋)



app/Http/Controllers/Auth

app/Http/Controllers/Requests/Auth

app/View/Components

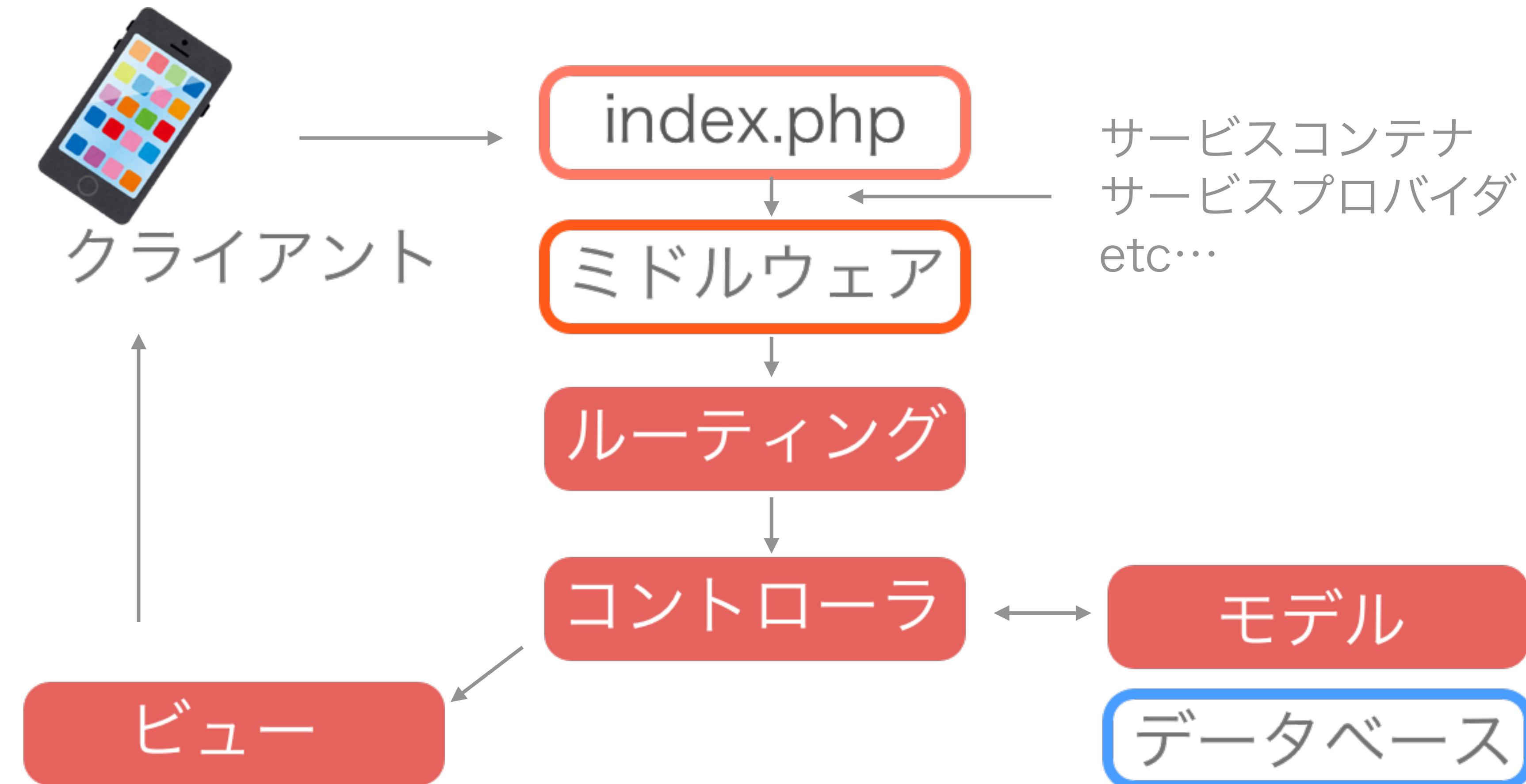
routes/web.php

routes/auth.php

resources/views/auth

resources/views/components

ブラウザに表示されるまでの流れ



MVCモデル: Model, View, Controller

認証機能、追加ファイル



サービスプロバイダ
config/app.php 内の
providers, aliases にAuthと記載

ルーティング
routes/web.php
routes/auth.php

ルートファイル

```
use Illuminate\Support\Facades\Route; //Routeを読み込む  
use App\Http\Controllers\Auth\RegisteredUserController; //コントローラを読み込む
```

```
Route::get('/register', //Route::getかpost (url)  
[RegisteredUserController::class, 'create']) //[]でコントローラ名, メソッド名  
->middleware('guest') //middleware guestだったら  
->name('register'); // 名前付きルート
```

マニュアル: 認証

->ルートの保護, リダイレクト, ガードの指定, ログイン回数制限

ルートファイル



現在のルーティング設定を確認するコマンド

```
php artisan route:list
```

テキスト出力

```
php artisan route:list > route.txt
```

コントローラファイル群

app/Http/Controllers/Auth

authenticatedSession //認証セッション
confirmablePassword //パスワード確認
emailVerificationNotification //メール検証通知
emailVerificationPrompt //メール検証プロンプト
newPassword //新しいパスワード
passwordResetLink //パスワードリセットリンク
registeredUser //ユーザー登録
verifyEmail //メール検証

ビューファイル群

Resources/views/auth

confirm-password //パスワード確認

forgot-password //パスワード忘れ

login //ログイン

register //新規登録

reset-password //パスワードリセット

verify-email //メール検証

Bladeファイルは xxx.blade.php の形式で作成する

Laravel Breeze 日本語化



resources/lang/ja フォルダ配下に
マニュアル下部の4つの言語ファイルを配置
page.php, pagination.php,
passwords.php, validation.php

* Laravel9 の場合は lang/ja フォルダ

Laravel Breeze 日本語化



resources/lang/ja/validation.phpを調整

```
'attributes' => [
    'name' => '名前', // 追加
    'email' => 'メールアドレス', // 追加
    'password' => 'パスワード' // 追加
],
```

Woops も日本語化するなら



resources/lang/ja.json を作成

```
{"Woops! Something went wrong.": "おっ  
と、サーバーで何か問題が発生しました。"}
```



Tailwindcss

tailwind(追い風)css



CSSフレームワーク

✗UIキット ○ユーティリティクラス
カスタマイズ(微調整)しやすい

Lowレベル(CSSに近い)
クラス名・・CSSプロパティ名に近い
->CSSも身につく
レスポンシブ対応, Flexbox, Grid採用

レスポンシブ対応



同じコードでPC, SP, タブレットを表示
ブラウザの幅に合わせて見やすく自動調整

モバイルファースト(インデックス)
2018年->2020年->2021年

<https://www.itra.co.jp/webmedia/what-is-mfi.html>

モバイルファースト



Breakpoint prefix	Minimum width	CSS
`sm`	640px	`@media (min-width: 640px) { ... }`
`md`	768px	`@media (min-width: 768px) { ... }`
`lg`	1024px	`@media (min-width: 1024px) { ... }`
`xl`	1280px	`@media (min-width: 1280px) { ... }`
`2xl`	1536px	`@media (min-width: 1536px) { ... }`

全ての幅で共通 -> md (タブレット) -> lg (ノートPC)
の順でつくる

レスポンシブ対応



bg-green-300 md:bg-blue-300 lg:bg-red-300
全ての幅->md以上->lg以上

md:flex で md以上でFlexbox有効

LaravelBreeze追記ファイル



package.json・・・インストールバージョン
tailwindcss, tailwindcss/forms

tailwind.config.js・・・ページ設定など
webpack.mix.js・・・requireで読み込み
に追記

詳しくは別講座にて



【tailwindcss】 CSSが苦手な方向け
じっくり取り組んでみよう



追加:
Tailwindcss ver3
変更にあたって

Tailwindcss ver3



2021年12月にリリース

Laravel Breezeでもver3インストールに変更

Just-In-Time機能

必要なものを、必要なときに、必要なだけ
->使ったクラスだけを出力する機能

Tailwindcss ver3



Tailwind.config.js

purge->contentに変更

ここに記載されているファイルで

Tailwindcssが使われていたら出力される

(public/css/app.cssに出力)

php artisan serveだけだと反映されない

npm run dev やnpm run watchで反映

Tailwindcss ver3



ターミナルを2つもしくは3つ開く

ターミナル1: npm run watch

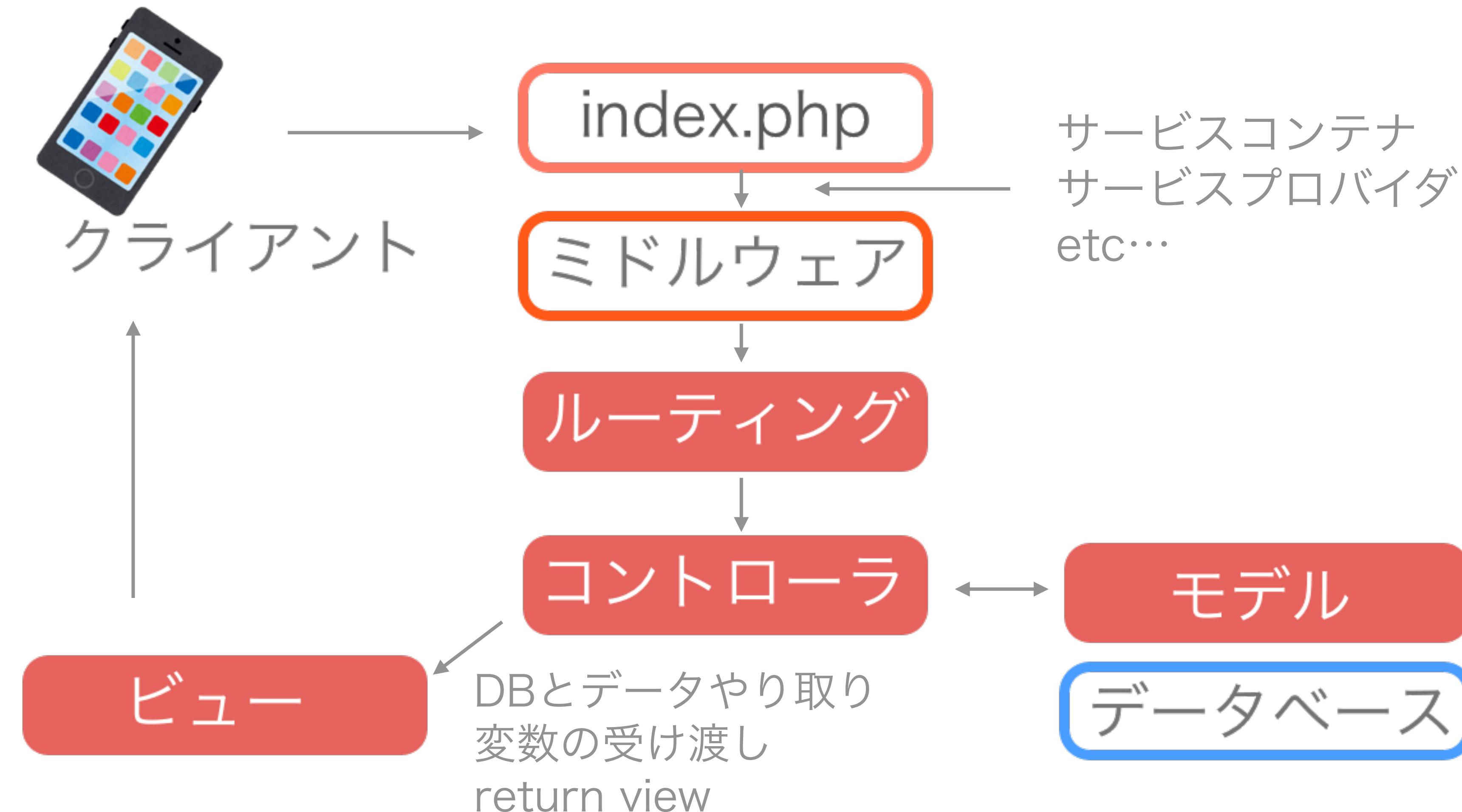
ターミナル2: php artisan serve

ターミナル3: Laravel各コマンド実行



Blade
コンポーネント

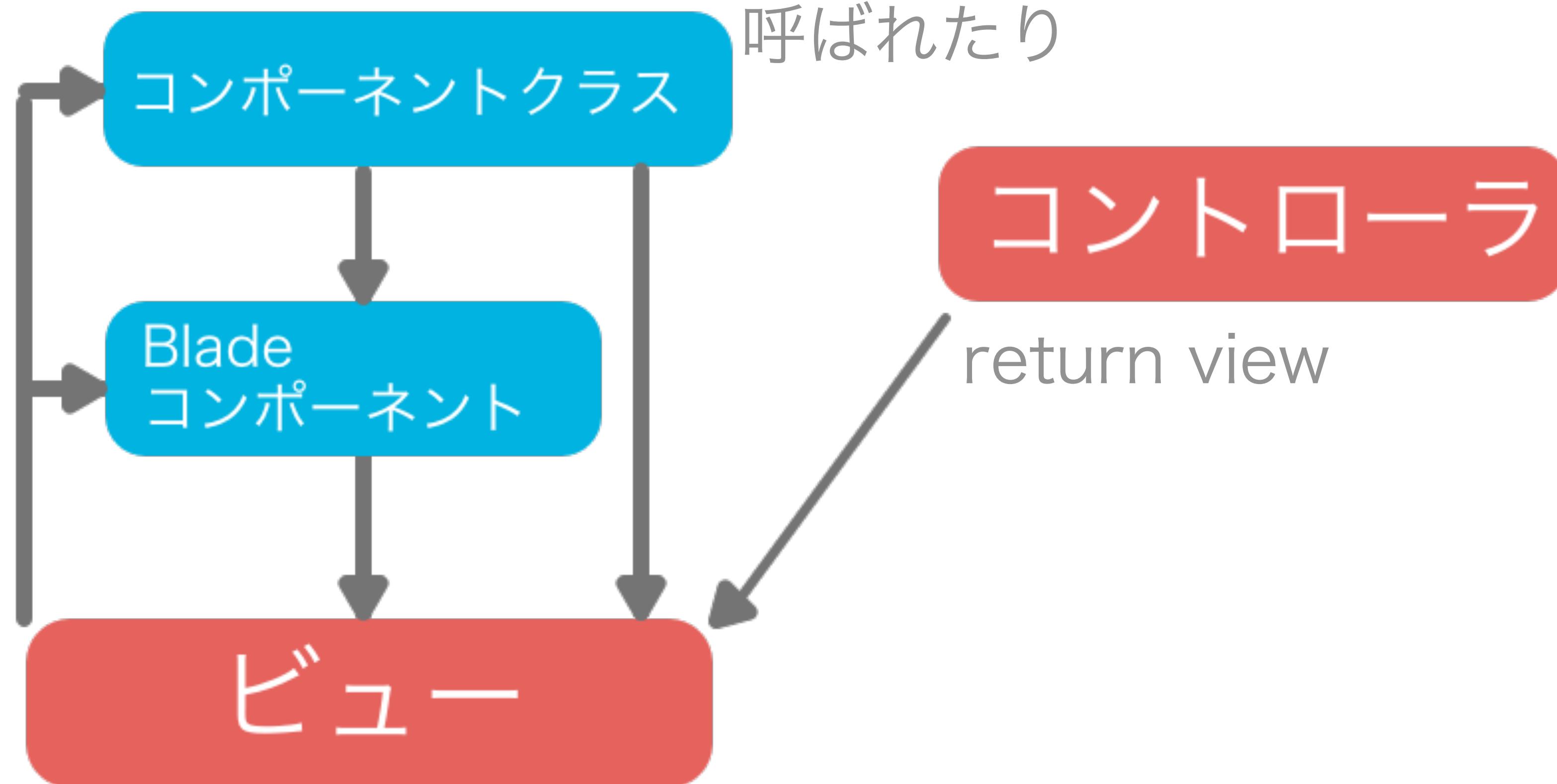
Controller -> View の役割



ControllerとViewの分離

DBとデータのやり取り
変数の受け渡し

サブコントローラーと
呼ばれたり



機能比較

	テンプレート継承	コンポーネント
バージョン		7.x～
用途	(共通)ヘッダー、サイドバー、フッター (個別)コンテンツ	共通部分の表示 データや属性の受け渡し スロット(差し込み)
メリット	PHPのrequireに似ている。シンプル	ControllerとViewの分離
ファイル	resources/views/xxx.blade.php	app/View/Components/xxx.php resources/views/components/xxx.blade.php resources/views/xxx.blade.php
関連タグなど	@yield, @extend, @section, @include	x-コンポーネント名 (ケバブケース) <x-alert type="error" :message="\$message">
関連機能 (スロット)		スロット {{ \$slot }} <x-alert>コンポーネント内の文字</x-alert>

Bladeコンポーネント

	生成方法	View/Component	resources/views/ components	resources/views/ components
クラスベース	<code>php artisan make:component xxx</code>	○	○	○
インライン	<code>php artisan make:component xxx --inline</code>	○		○
匿名コンポーネント	直接ファイル作成		○	○

BladeComponentの準備



ルーティング

routes/web.php

useでコントローラ読み込み、Route::でルーティング設定

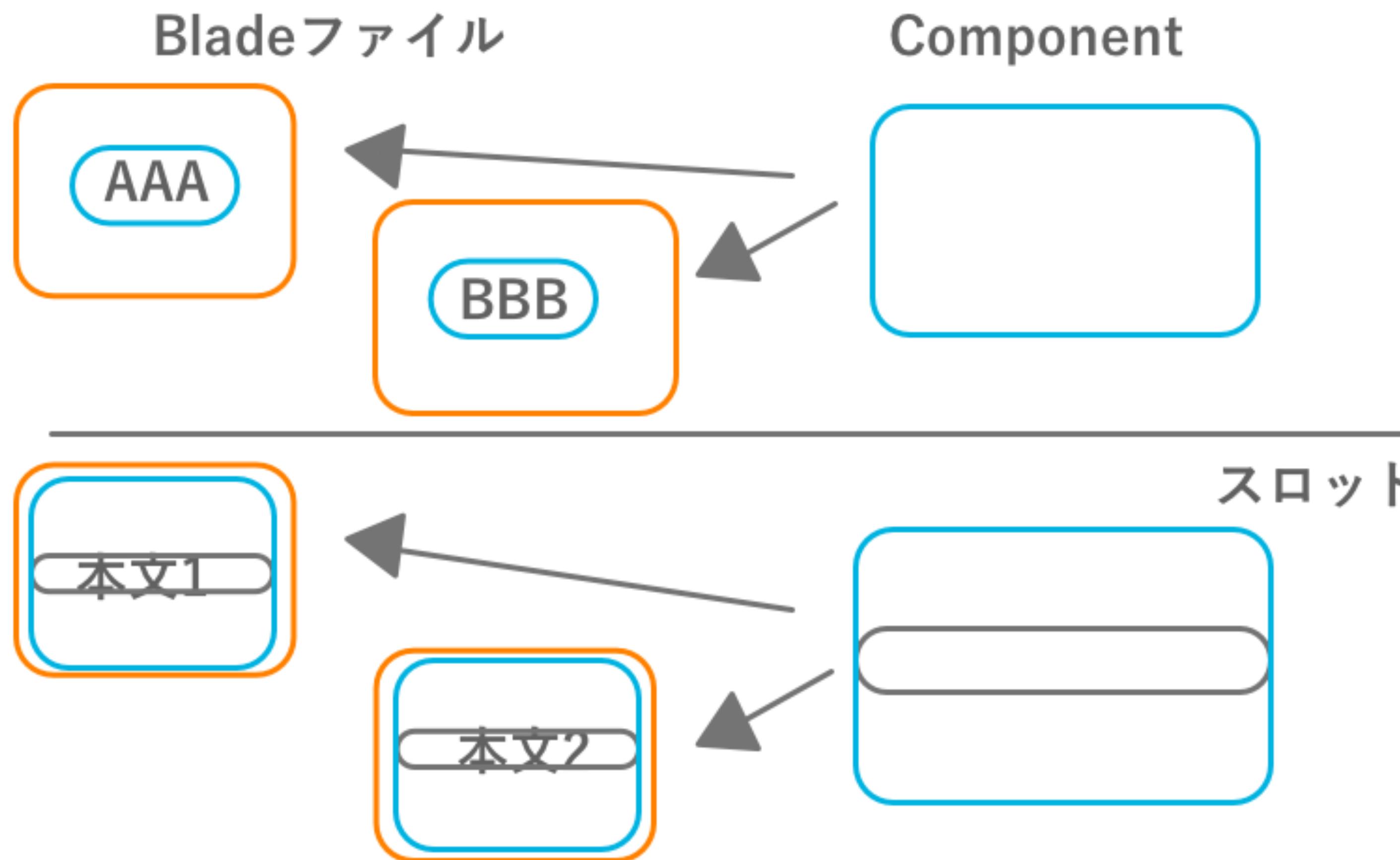
コントローラ

php artisan make:controller ComponentTestController
でファイル作成

```
public function showComponent1(){
    return view('tests.component-test1');
}
```

Componentのパターン

1つのコンポーネント(部品)を複数ページで使える
コンポーネント側を修正すると全て反映される



Componentの書き方



resources/views/components フォルダ内に配置

<x-コンポーネント名></x-コンポーネント名>

フォルダで分けたい場合

resources/views/components/tests フォルダの場合

<x-tests.コンポーネント名></x-tests.コンポーネント名>

Slot



Component側

```
{{ $slot }}
```

-> {} マスタッシュ構文 (口ひげ)

Blade側

```
<x-app>この文章が差し込まれる</x-app>
```

名前付きSlot



Component側
{{ \$header }}

Blade側
<x-slot name="header">この文章が差し込まれる
</x-slot>

BladeファイルとComponent

	Bladeファイル	Bladeコンポーネント
\$slot	<x-app>ここに文字</x-app>	{{ \$slot }}
名前付きslot	<x-slot name="header">ここに文字</x-slot>	{{ \$header }}
属性 (props)	<x-card title="タイトル">	{{ \$title }}
変数	<x-card :message="\$message">	コントローラなどに指定 \$message {{ \$message }}
初期値 @props 連想配列	設定しない場合 初期値が表示される	@props(['message' => '初期値です。'])
クラスの設定 属性バッグ	class="bg-red-300"	<div {{ \$attributes->merge(['class' => 'text-sm']) }}> </div>

コントローラなどから変数を渡す

コントローラ側

```
$message = 'メッセージ';
```

```
return view('ビューファイル', compact('message'));
```

Blade側

```
<x-card :message="$message" />
```

```
<x-card :コンポーネントで指定した変数名=コントローラで指定した変数名 />
```



クラスベースの
コンポーネント

クラスベースのコンポーネント



App/View/Components内のクラスを指定する

クラス名 · · TestClassBase (パスカルケース)
Blade内 · · x-test-class-base(ケバブケース)

コンポーネントクラス内で

```
public function render(){
    return view('bladeコンポーネント名')
}
```

匿名とクラスベースの比較表

	Bladeファイル	Bladeコンポーネント(匿名)	クラスベース
\$slot	<x-app>ここに文字</x-app>	{{ \$slot }}	同左
名前付き slot	<x-slot name="header">ここに文字</x-slot>	{{ \$header }}	同左
属性 (props)	<x-card title="タイトル">	{{ \$title }}	public \$title public function __construct(\$title){ \$this->title = \$title }
変数	<x-card :message="\$message">	コントローラなどに指定 \$message {{ \$message }}	クラス内に指定可
初期値 @props 連想配列	設定しない場合 初期値が表示される	@props(['message' => '初期値です。'])	public function __construct(\$title="初期値"){ \$this->title = \$title }
クラスの設定 属性バッグ	class="bg-red-300"	<div {{ \$attributes->merge(['class' => 'text-sm']) }}> </div>	同左

クラスベースで属性などを扱う

```
Class TestClassBase
{
    public $classBaseMessage;
    public $defaultMessage;

    public __construct($classBaseMessage, $defaultMessage="初期値")
    {
        $this->classBaseMessage = $classMessage;
        $this->defaultMessage = $defaultMessage;
    }
}
```

LaravelBreezeで使われているコード



`{{ __() }}` 多言語用 言語ファイルから文字列取得

`{{ $value ?? $slot }}`

Null合体演算子(isset+三項演算子) nullでなければ\$valueを返す

`{!! !!}` エスケープしないデータの表示

`@php ~ @endphp` php実行

マニュアルも参考に



Laravel 8.x Bladeテンプレート

<https://readouble.com/laravel/8.x/ja/blade.html>

View側のキャッシュを削除するコマンド

`php artisan view:clear`

Alpine.js

Alpine.js



公式GitHub

<https://github.com/alpinejs/alpine>

tailwindのJavaScript版のようなもの(軽量)

タグ内に専用のディレクティブを設置できる。

<div x-show="isOpen()"></div>

Alpine.jsとVue.js

Alpine.js	Vue.js	特徴
x-data	dataプロパティ	データの状態 オブジェクトでも書ける
x-init	mounted() フック	DOM更新時に実行
x-show	v-show	Trueなら表示
x-bind:属性="式", :属性="式"	v-bind	属性の値を設定
x-on:click="", @click=""	v-on	イベント時のメソッドなどを設定
x-model	v-model	双方向データバインディング
x-text, x-html	v-text, v-html	テキスト表示、HTML表示
x-ref		コンポーネントからDOM取得
x-if, x-for	v-if, v-for	if文、for文
x-transition	v-transition	トランジション
x-spread		再利用できるオブジェクトに抽出
x-cloak	v-cloak	チラつき防止

Vue.jsの講座も参考に

【Vue.js】じっくりとことんやってみよう
【超初心者から脱初心者へレベルアップ】

