

Laravel

わかりやすさ重視 編



ヘルパ関数

Laravelが用意している便利関数



その中でも特によく使う

route, auth, app, bcrypt

collect, dd, env, factory, old, view, など

[https://readouble.com/laravel/6.x/ja/
helpers.html](https://readouble.com/laravel/6.x/ja/helpers.html)



コレクション型

配列を拡張した型



データベースからデータ取得時は

コレクション型になっている

コレクション型専用の関数多数

メソッドチェーンで記述可能

all, chunk, get, pluck, whereln, toArray

<https://readouble.com/laravel/6.x/ja/collections.html>



クエリビルダ

クエリをPHPでかける



Select, where, groupbyなど

SQLに近い構文

DB::table(テーブル名)-> とつなぐ

[https://readouble.com/laravel/6.x/ja/
queries.html](https://readouble.com/laravel/6.x/ja/queries.html)



ファード

Facade 正面入り口



Select, where, groupbyなど

SQLに近い構文

DB::table(テーブル名)-> とつなぐ

[https://readouble.com/laravel/6.x/ja/
facades.html](https://readouble.com/laravel/6.x/ja/facades.html)



Laravel起動処理

DIと

サービスコンテナ

Laravel起動まで



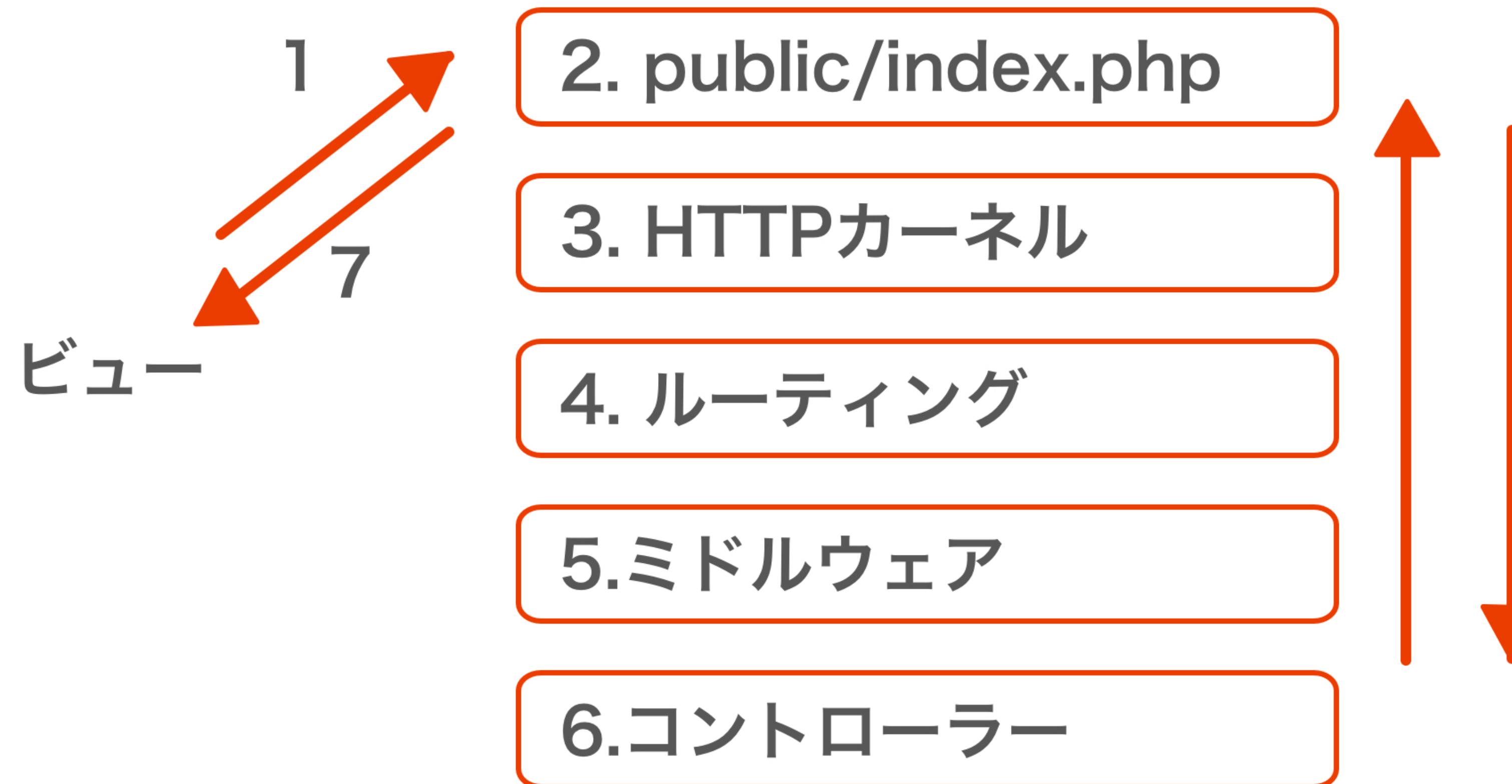
DI・・外部でインスタンス化して注入

サービスコンテナ・・DIをまとめて担う

<https://qiita.com/namizatop/items/801da1d03dc322fad70c>

[801da1d03dc322fad70c](https://qiita.com/namizatop/items/801da1d03dc322fad70c)

Laravel起動・表示の流れ





Blade

Blade テンプレートエンジン



`{{ }}` でエスケープ処理して表示

`@csrf` でCSRF対策

`@foreach @endforeach` で配列表示

`@section @yield`などでテンプレート読み込み

<https://readouble.com/laravel/6.x/ja/>

blade.html



フロントエンド (FrontEnd)

フロントエンドとバックエンド

クライアントサイドとサーバーサイド



クライアント



サーバー



ニーズを求めた結果・・



クライアント・・使いやすい・表示が早い

開発側・・作りやすい・管理しやすい

どんどんカオスな状態に



クライアント

HTML



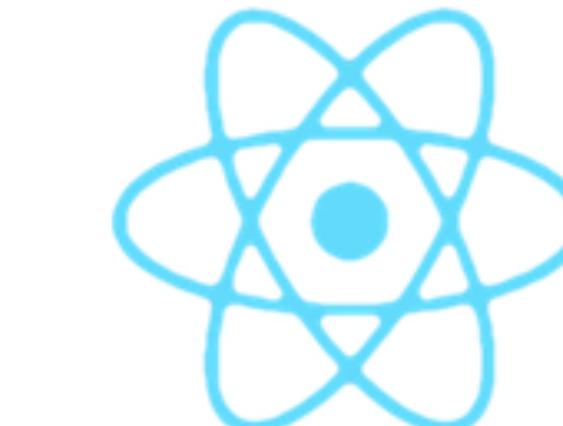
pug



Sass

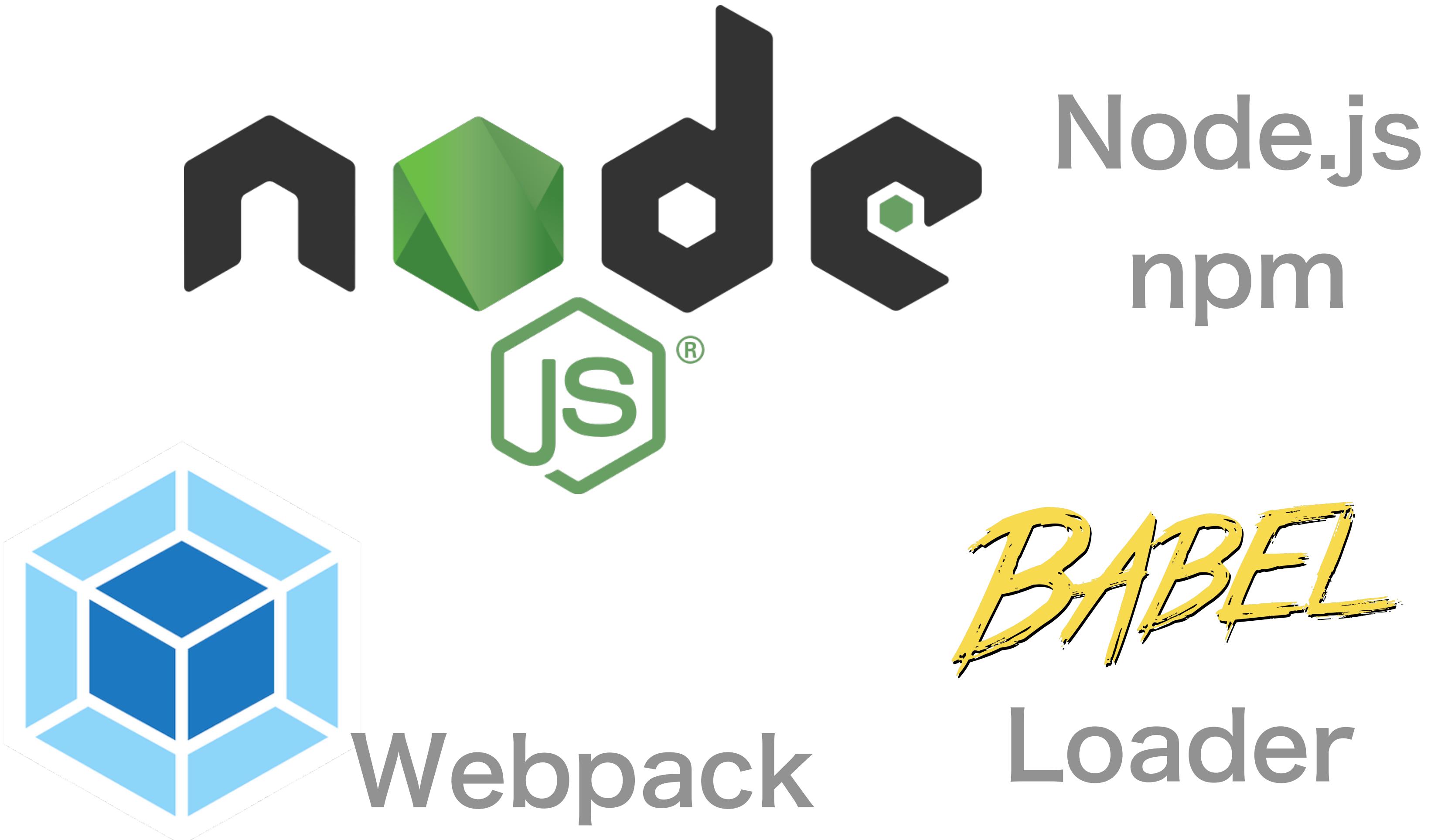


Vue.js

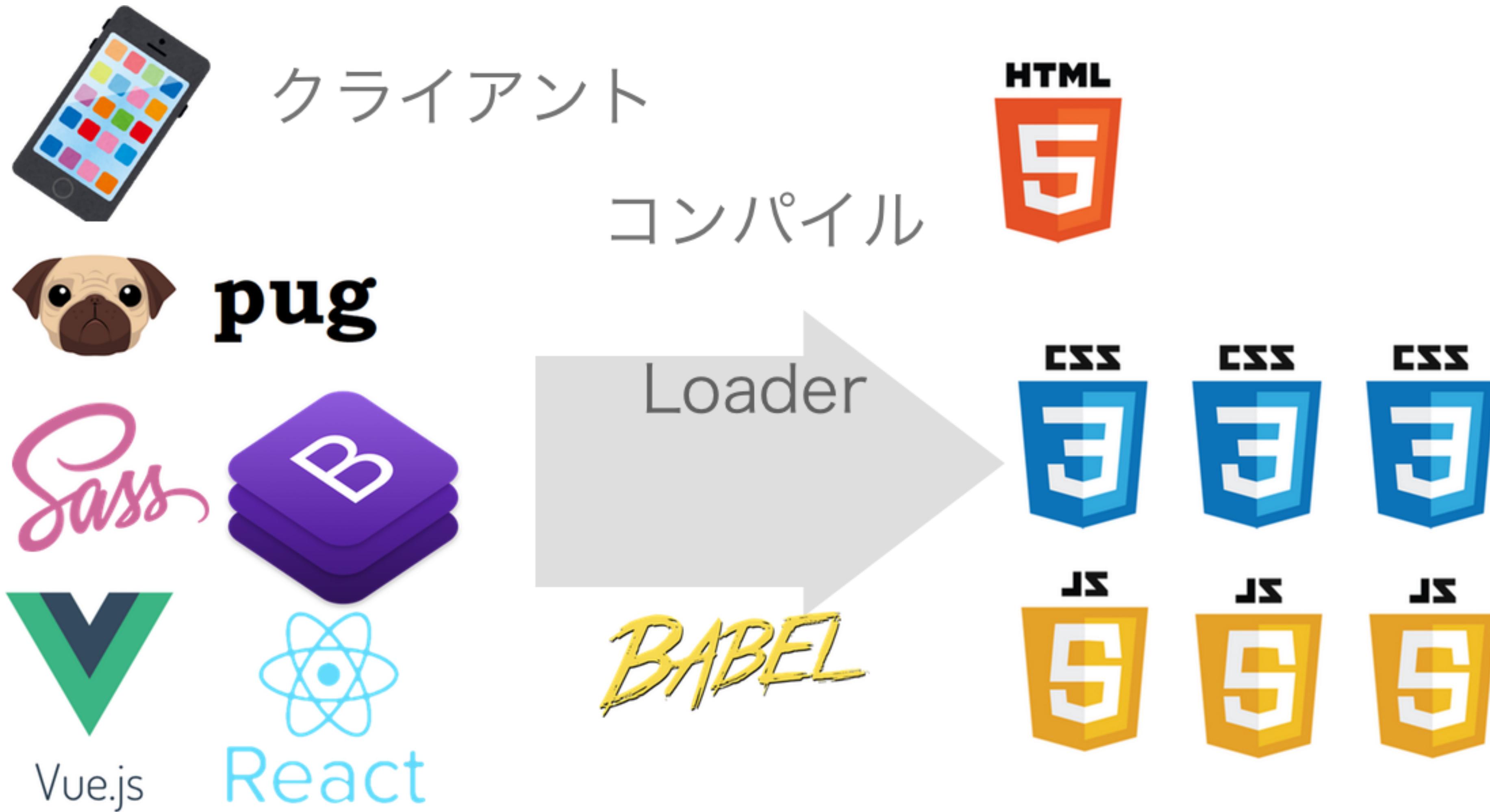


React

フロントエンド必須ツール



コンパイル(compile) (新->旧)



バンドル(bundle) (複数->1つ)



Laravelで設定必要



laravel-ui • • Laravel6.xから

laravel-mix • • webpackのラッパー

webpack.mix.js • • laravel-mixの設定ファイル

Node.js/npm • • 別途インストール

package.json / package-lock.json 設定管理ファイル



Laravel-ui

認証

laravel-uiインストールしてnpm



composer require laravel/ui --dev

php artisan ui bootstrap --auth

(他にvueやreactも選択可能)

npm install

npm run dev

<https://readouble.com/laravel/6.x/ja/frontend.html>

認証(Auth)の補足



エラーメッセージの日本語化

<https://github.com/minoryorg/laravel-resources-lang-ja>

php artisan route:list でリスト表示

マルチログイン機能もつくれる

<https://coinbaby8.com/laravel-multi-login.html>



簡易Webアプリ

簡易Webアプリ



お問い合わせフォーム拡張版(CRUD)

REST (RESTful を活用)

バリデーション

ダミーデータ(シーダー/ファクトリー)

ページネーション

簡易検索機能

Model ->Controller -> Route -> View

モデル・マイグレーション



よく使うカラム修飾子

`nullable()`, `unique()`, `unsigned()`

マイグレーションは

テーブルの履歴を管理できる仕組み

`migrate:rollback` などで戻る事ができる

RESTful コントローラー

よく使うメソッドをまとめて生成できる

php artisan make:controller ** —resource

動詞	URI	アクション	ルート名
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

ルーティング



よく使う機能

グループ機能 `Route::group`

`prefix` (共通のフォルダを指定)

`'middleware' => 'auth'`

(マルチログインの場合はそれぞれ作成必要)

`id`などで切り分ける `Route::get('show/{id}',);`



Viewファイルの 読み込み

auth/login.blade.php



@extends('layouts.app') で読み込み

@section('*') で中身

フォームには@csrfが必須

ルートにnameをつけておくと

{{ route('login') }} などでルーティング可

layouts/app.blade.php



ヘッダやフッターなど共通部分を使う

head内にcsrf-token必須

{()} 内にphp書ける

個別箇所は @yield('content') などで

{[--]} でコメントアウト



フォーム関連

フォーム関連



基本はPHP版と同じ。

formタグのactionは{{ route('name') }}で書ける

流れは

ルート→コントローラー→ビュー

Formファサードをインストールする方法もある

データベースからのデータ取得



新規登録や全件取得ならEloquent。

```
Contact::all();
```

表示する列を絞るならクエリビルダ。

```
DB::table('contact')->select('*')->get();
```

フォームはRequestで受け取る



public function store(Request \$request)

storeメソッドの引数は

DI(Requestインスタンスが入ってくる)

->メソッドインジェクションと呼ばれることがあります。

データベースからの情報取得



新規登録なら

```
$contact = new ContactForm;
```

1件取得なら (引数に\$idが必要)

```
$contact = ContactForm::find($id);
```

データベース変更後は redirect で画面遷移する



フォームリクエスト (バリデーション)

Laravel専用のバリデーション



フォームリクエストを使う

authorize メソッドはtrueに

uniqueを付ける場合はテーブル名も指定

unique:users など

コントローラ側はuseで読み込み

storeの引数にフォームリクエストを指定



ダミーデータ

ダミーデータ(テストデータ)



1つずつつくるならシーダー(seeder)

```
php artisan make:seeder UsersTableSeeder
```

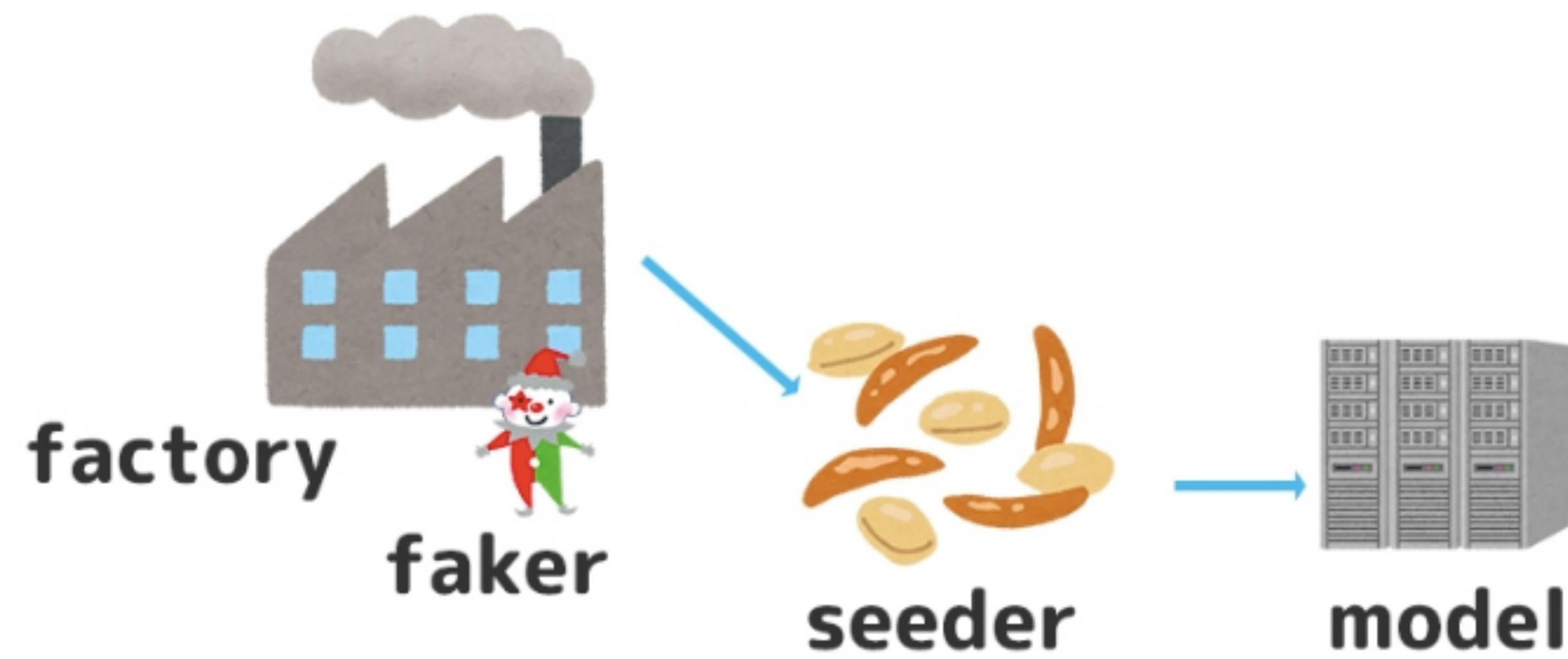
大量に作るなら

フェイカー(Faker) / ファクトリー(Factory)

も組み合わせる

大量のダミーデータ

factoryの中で fakerでダミーデータ



<https://coinbaby8.com/laravel-seeder-faker.html>



検索フォーム

検索フォーム



条件指定は where で

複数キーワードがあれば
空白などで切り離してそれぞれ
where句を追加する

他にも orWhere などの構文もあるので
クエリビルダのマニュアルを要チェック



要件定義と 基本設計

要件定義・基本設計(デザイン)



誰が何をどう使いたいか->抽象から具体へ

画面設計・・View

機能設計・・Controller

データ設計・・Model

<https://qiita.com/Saku731/items/741fcf0f40dd989ee4f8>

要件定義・基本設計



画面設計 (UI/UX)

Excel, Cacco, AdobeXD, Sketch(mac), Figma などなど

機能設計 (クラス図など)

Excel, draw.io

データ設計 (ER図など)

Excel, draw.io VS Code(UML)

検索システムサンプル

もんプロ 不動産検索システムサンプル

Home 新規作成 設定項目 お客様使用画面 test@test.com Logout

不動産検索システム

ID	賃貸/売買	物件名	表示/非表示	売出中/完売	プレゼント	登録日時
6	賃貸	東都大学ハイツ	表示	売出中	-	2020-01-22 14:46:57
5	おすすめ売買	おすすめだよ！	表示	売出中	-	2019-03-26 20:37:51
4	売買	あああ	表示	売出中	-	2019-03-26 20:36:46
3	賃貸	テスト	表示	売出中	-	2019-03-26 20:30:50
2	賃貸	てうと	表示	売出中	-	2019-03-25 23:19:22
1	賃貸	テスト	表示	売出中	-	2019-03-25 22:37:25

<https://coinbaby8.com/search-system.html>



リレーション

DBテーブル同士の関係



1対1・・あまり使わない

1対多(n)・・複数の中から1つを選べる(お店と住所)

多対多・・複数の条件を紐づける(お店と路線)

SQLだとJOINで紐づける

```
SELECT * FROM `shops`
```

```
INNER JOIN `areas`
```

```
ON `shops`.`area_id` = `areas`.`id`
```

ER図に少しずつ慣れる



PK・・主キー、プライマリーキー(Primary)

FK・・外部キー制約(Foreign)

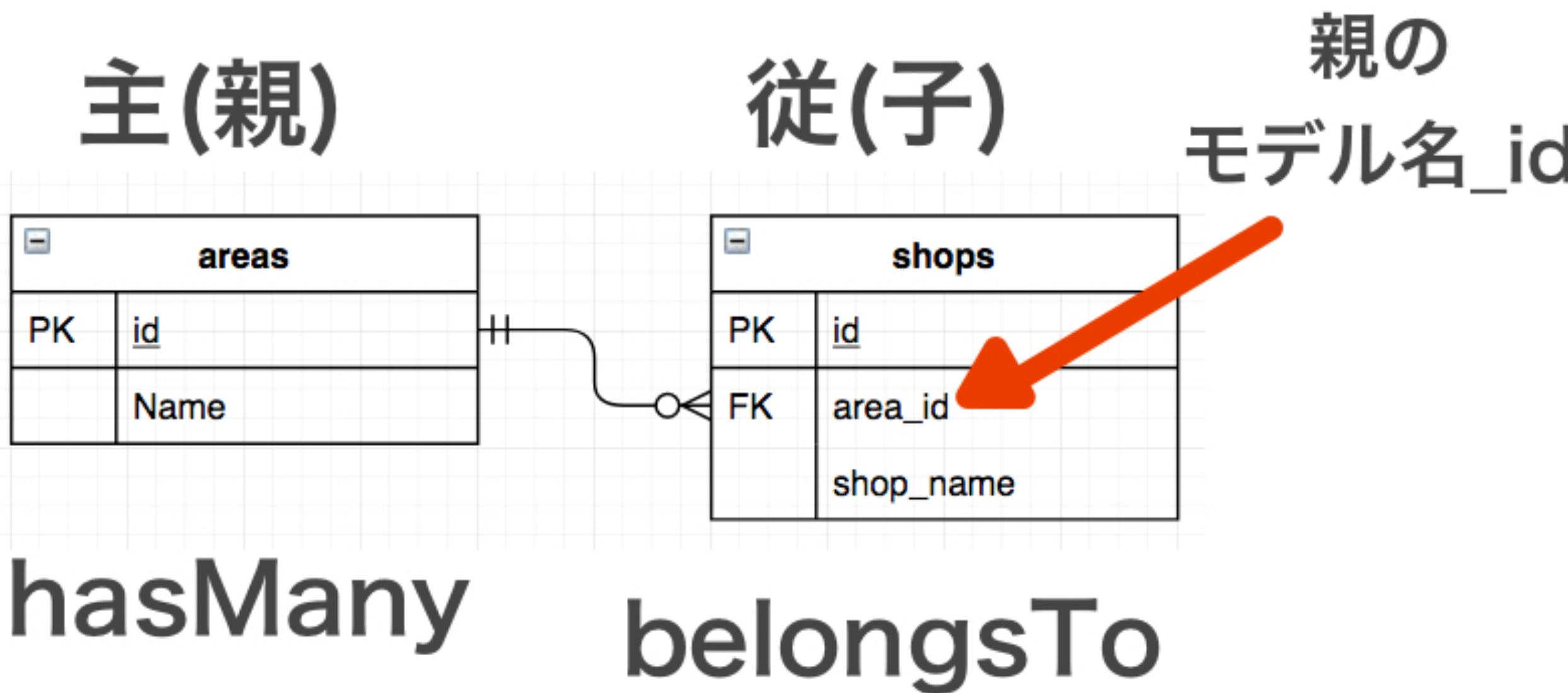
(それぞれ複数の設定も可能)

<https://it-koala.com/entity-relationship-diagram-1897>

DBテーブル構成:第3正規化まで必須

<https://qiita.com/mochichoco/items/2904384b2856db2bf46c>

リレーション 1対多 (1対n)



モデルにhasMany / belongsTo を追加
親のモデル名_id にしておくと楽

1対n 外部キー設定時の注意



FK設定の場合、
主テーブルの内容がないと
従テーブルに値を設定できない

必ず主テーブルから設定する
(マイグレーション、シーダー)

リレーション 多対多 (n対n)

中間テーブル(pivot table)を作成



PKをid以外にする場合は

マイグレーションにprimaryを追記

モデル・・・ belongsToMany



おまけ

ファイルアップロード方法



昔 FTP

最近 Git/GitHub (SSH · · Secure Shell)

<https://git-scm.com/>

<https://github.com/>

Gitの解説 <https://zukulog098r.com/git/>

.gitignore · · Gitに含めないファイル

サーバーの種類



レンタルサーバー・・・

Xserver/さくら/ロリポップ etc

VPS・・さくら/Konoha/KAGOYA

PaaS・・Heroku

クラウド・・AWS, GCP, Azure

サンプルプロダクト Koel



Koel <https://github.com/phanan/koel>

git clone ***

php artisan key:generate

.env 編集 / php artisan migrate

composer install / npm install

npm run dev / npm run prod

デプロイ (アップロード方法)



今回はXサーバーをサンプルに

ローカルPC -> GitHub -> Xサーバー

git push / git pull

SSH (公開鍵・秘密鍵)

サーバー側に必要

php7.2以上、composer, git, (node.js)