




Laravel9 その3



サービスへの 切り離し

処理の分離

全ての処理をコントローラに書いていくと

コントローラがどんどん肥大化する

(ファットコントローラと呼ばれる)

処理を別ファイルに切り分ける事で

コントローラをできるだけスリムにする

ex) 処理->コントローラ

ex) バリデーション->カスタムリクエスト

ex) データベース処理->クエリスコープ

サービス作成

App/Services/CheckFormService.php

```
<?php
```

```
namespace App\Services;
```

```
class CheckFormService
```

```
{
```

```
    public static function checkGender($data){
```

```
        if( $data->gender === 0 ){ $gender = '男性'; }
```

```
        if( $data->gender === 1 ){ $gender = '女性'; }
```

```
        return $gender;
```

```
    }
```

```
    public static function checkAge($data){
```

```
        if ( $data->age === 1 ){ $age = '～19歳'; }
```

```
        略 return $age;
```

```
    }
```

```
}
```

コントローラで読み込み

ContactFormController.php

```
use App\Services\CheckFormService;

public function show($id)
{
    // staticで指定しているので👤でメソッドを指定できる
    $gender = CheckFormService::checkGender($contact);
    $age = CheckFormService::checkAge($contact);

    return view('contacts.show', compact('contact', 'gender', 'age'));
}
```




フォームリクエスト (バリデーション)

フォームリクエスト 1

バリデーションはコントローラにも書けるけれど
フォームリクエストを使うとファイル分離してすっきり書ける

```
php artisan make:request StoreContactRequest
```

```
App/Http/Requests/StoreContactRequest.php
```

```
public function authorize(){  
    return true; // trueにしないと弾かれる  
}  
  
public function rules()  
{ 次ページ }
```


フォームリクエスト 2

```
public function rules()
{
    return [
        'name' => ['required', 'string', 'max:20'],
        'title' => ['required', 'string', 'max:50'],
        'email' => ['required', 'email', 'max:255'], //テーブル毎に1件ならunique:contact_forms
        'url' => ['url', 'nullable'],
        'gender' => ['required', 'boolean'],
        'age' => ['required'],
        'contact' => ['required', 'string', 'max:200'],
        'caution' => ['required', 'accepted']
    ];
}
```


コントローラで指定

コントローラでuseで読み込んで

storeメソッドやupdateメソッドの引数に設定する

ContactFormController.php

```
use App\Http\Requests\StoreContactRequest;
```

```
public function store(StoreContactRequest $request)
```

```
{
```

略

```
}
```


ビュー側

resources/views/auth/login.blade.php

の<x-auth-validation-errors>をコピー

resources/views/contacts.blade.php

に追加

エラーの日本語対応するならlang/ja/validation.phpの
attributesに追記

oldヘルパ関数

バリデーションで弾かれると入力していた情報が消えてしまう

old() とつけることでバリデーション後も入力値を保持できる

valueにold(name属性)をつける

```
value="{{ old('name') }}"
```

```
value="{{ old('title') }}"
```

radioとselectは三項演算子で書くとシンプルに書ける

```
radio {{ old('gender') == '1' ? 'checked' : "" }}
```

```
select {{ old('age') == '1' ? 'selected' : "" }}
```




ダミーデータ

ダミーデータ(テストデータ)

1つずつ作るならシーダー(seeder)に直書き

```
php artisan make:seeder TestSeeder
```

시더 파일 생성 명령어

```
php artisan make:seeder UserSeeder
```

database/seeds/TestSeeder.php

```
use Illuminate\Support\Facades\DB;
```

```
use Illuminate\Support\Facades\Hash;
```

시더 파일에 반드시 입력 해줘야 한다

// 複数投入する場合は[] の中に [] を書く必要があるので注意

```
public function run(){
```

```
    DB::table('tests')->insert([
```

```
        ['text' => 'aaa', 'created_at' => Now()]
```

```
        ['text' => 'bbb', 'created_at' => Now()]
```

```
    ])
```

ダミーデータ(テストデータ)

database/seeds/UserSeeder.php

```
use Illuminate\Support\Facades\DB;
```

```
use Illuminate\Support\Facades\Hash;
```

```
public function run(){
```

```
    DB::table('users')->insert([
```

```
        'name' => 'aaa',
```

```
        'email' => 'test@test.com',
```

```
        'password' => Hash::make('password123'); //パスワードは暗号化が必要
```

```
    ]);
```

```
}
```


ダミーデータ(テストデータ)

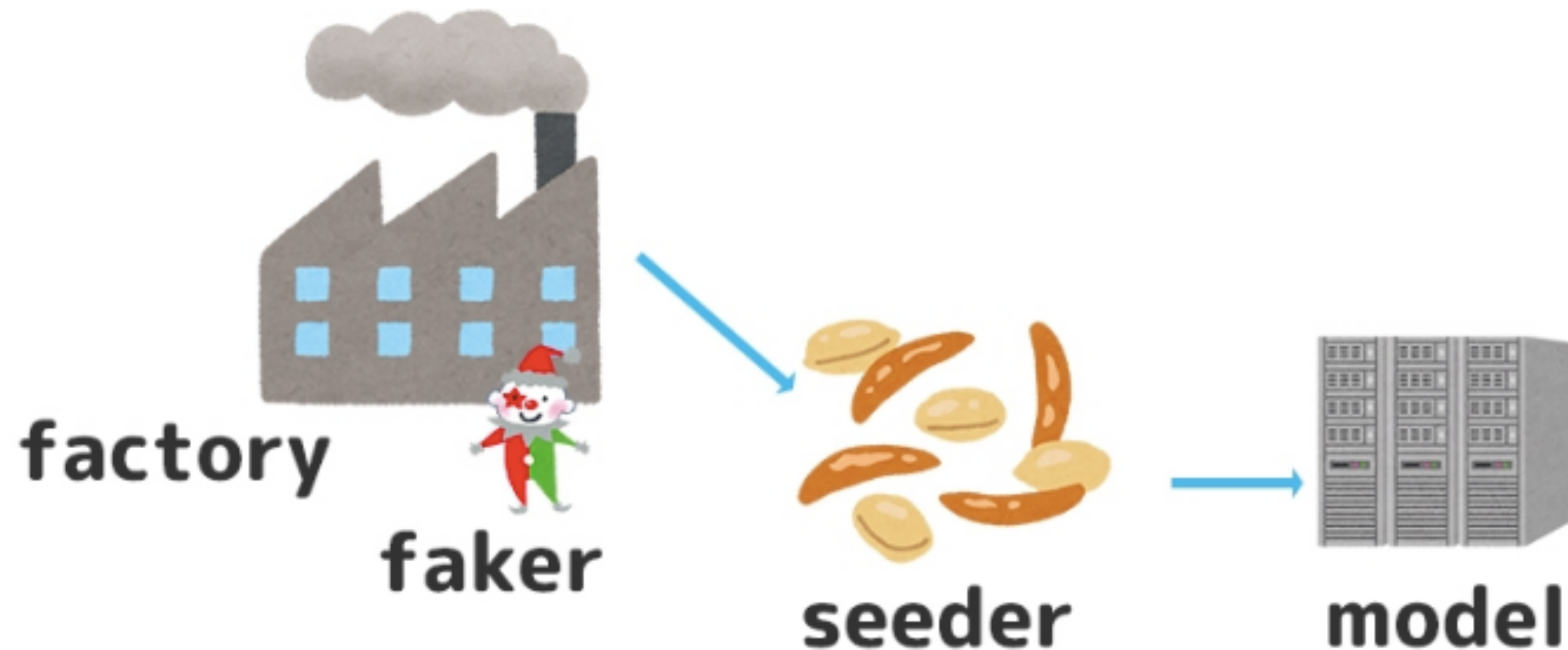
登録したダミーデータを `database/seeds/DatabaseSeeder.php` に追記

```
class DatabaseSeeder
{
    public function run(){
        $this->call([
            UserSeeder::class,
            TestSeeder::class ]); } }
```

設定後、`php artisan migrate:fresh --seed` でテーブル再作成時にダミーも投入

大量のダミーデータ

factoryの中で fakerを使ってダミーデータをつくる



Factoryの中でfakerを使う

`php artisan make:factory ContactFormFactory` 팩토리 생성 명령어

参考URL <https://qiita.com/tosite0345/items/1d47961947a6770053af>

database/factories/**ContactFormFactory.php**

```
public function definition()
```

```
{ return [
```

```
    'name' => $this->faker->name(20),
```

```
    'title' => $this->faker->realText(50),
```

```
    'email' => $this->faker->email(),
```

```
    'url' => $this->faker->url(),
```

```
    'gender' => $this->faker->boolean(),
```

```
    'age' => $this->faker->numberBetween(1, 6),
```

```
    'contact' => $this->faker->realText(200),
```

```
];}
```

ダミーデータ (テストデータ)

config/app.php

faker_locale => 'ja_JP',

database/seeder/DatabaseSeeder.php

\App\Models\ContactForm::factory(100)

->create();

php artisan migrate:fresh --seed でダミー投入



ペジネーション

ペジネーション

大量のデータがあった際に

ページ毎に○件と指定して表示する機能

ContactFormController.php@index

```
$contacts = ContactForm::略->paginate(20);
```

views/contacts/index.blade.php

```
{{ $contacts->links() }}
```

ペジネーションのカスタマイズ

```
php artisan vendor:publish --tag=laravel-  
pagination
```

resources/views/vendor/paginationが生成

tailwind.blade.phpを編集して日本語化対応など



検索フォーム

検索フォーム

条件指定は `where` で `like` を使う事であり検索ができる

%は0文字以上

```
select * from `contact_forms` where `name` like '%aaa%'
```

AND検索

```
select * from `contact_forms` where `name` like '%aaa%' and `name` like  
`%bbb%`
```

Contacts/index.blade.php

```
<form method="get" action="{{ route('contacts.index') }}">
```

```
  <input type="text" name="search" placeholder="検索">
```

```
  <button>検索する</button>
```

```
</form>
```


コントローラ

ContactFormController.php

```
public function index(Request $request)
{
    $search = $request->search;
    $query = ContactForm::search($search); //クエリのローカルスコープ

    $contacts = $query->select . . . 略;


    return view('contacts.index', compact('contacts'));
}
```

クエリのローカルスコープ

App/Models/ContactForm.php

// メソッド名先頭にscope。第一引数は\$query、第二引数に渡ってくる引数

```
public function scopeSearch($query, $search)
{
    if($search !== null){
        $search_split = mb_convert_kana($search, 's'); // 全角スペースを半角
        $search_split2 = preg_split('/[\s]+/', $search_split); //空白で区切る
        foreach( $search_split2 as $value ){
            $query->where('name', 'like', '%' . $value. '%'); }
        }
    return $query;
}
}
```

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な→抽象から具体へ

画面設計・・・View

機能設計・・・Controller

データ設計・・・Model

参考記事

[https://qiita.com/Saku731/items/
741fcf0f40dd989ee4f8](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)

※ 2022年現在 draw.io は app.diagrams.netというURLに変更になっています。(リダイレクトがかかります。)


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>

An abstract graphic on the left side of the page, featuring bold, expressive brushstrokes in a variety of colors including yellow, orange, red, pink, purple, and blue. The strokes are layered and textured, creating a dynamic and energetic visual effect.

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つを選ぶ(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)

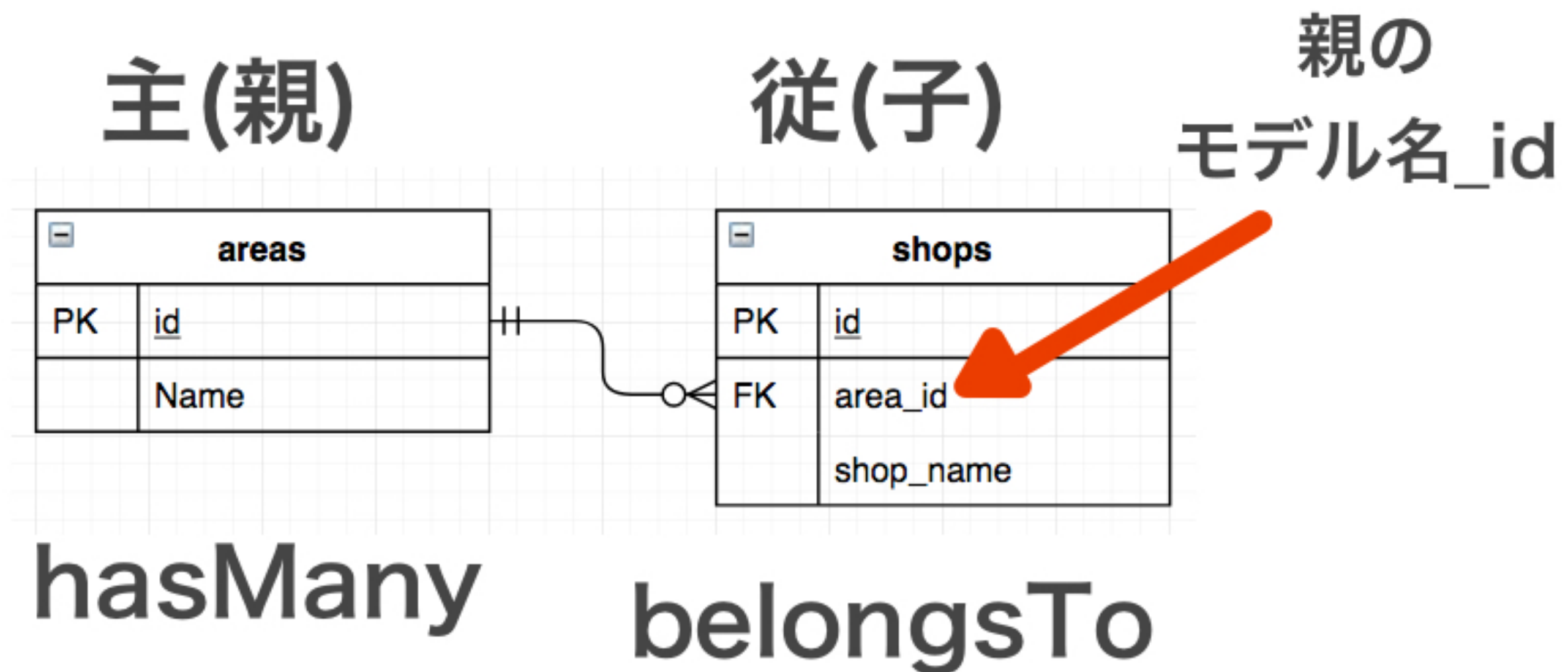
(それぞれ複数の設定も可能)

<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** / **belongsToMany** メソッドを追加

親のモデル名_id にしておくこと楽

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)

モデル・マイグレーション・ダミー生成

// エリア

```
php artisan make:model Area -ms
```

// 路線 (次回の多対多で使用)

```
php artisan make:model Route -ms
```

// 店舗

```
php artisan make:model Shop -ms
```


マイグレーション

create_areas_table.php (親)

```
Schema::create('areas', function (Blueprint $table) {  
    $table->id();  
    $table->string('name', 20);  
    $table->integer('sort_no');  
    $table->timestamps();  
});
```

create_shops_table.php (子)

```
Schema::create('shops', function (Blueprint $table) {  
    $table->id();  
    $table->string('name', 20);  
    $table->foreignId('area_id'); // 親のidと紐づく列 モデル名_id だと追加設定不要  
    $table->timestamps();  
});
```

AreaSeeder

```
use Illuminate\Support\Facades\DB;

public function run()
{
    DB::table('areas')->insert([
        [ 'name' => '東京', 'sort_no' => 1 ],
        [ 'name' => '大阪', 'sort_no' => 2 ],
        [ 'name' => '福岡', 'sort_no' => 3 ]
    ]);
}
```


ShopSeeder

```
use Illuminate\Support\Facades\DB;

public function run()
{
    DB::table('shops')->insert([
        [ 'name' => '高級食パン屋', 'area_id' => 1 ],
        [ 'name' => '高級クロワッサン屋', 'area_id' => 2 ],
        [ 'name' => '高級コッペパン屋', 'area_id' => 1 ],
        [ 'name' => '高級メロンパン屋', 'area_id' => 3 ]
    ]);
}
```

DatabaseSeeder

```
public function run()
{
    $this->call([
        UserSeeder::class,
        AreaSeeder::class, // AreaSeederを先に投入
        ShopSeeder::class,
    ]);
}
```

php artisan migrate:fresh --seed でダミー投入

JOIN

phpMyAdminでjoin確認してみる

```
select * from `shops`  
inner join `areas`  
on `shops`.`area_id` = `areas`.`id`
```

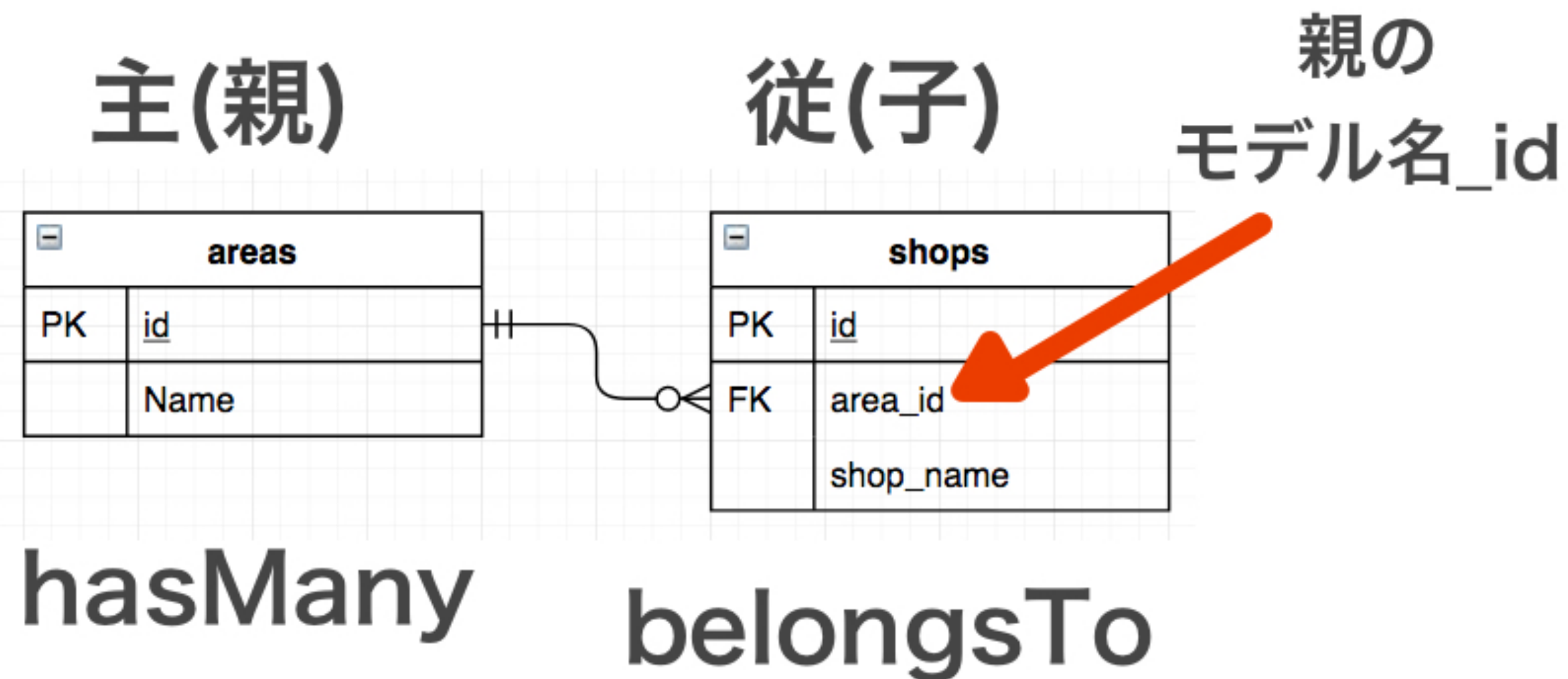
Laravelのクエリビルダにもjoinがある

エロクアントならリレーションを簡単に書ける



リレーション(1対多)

リレーション 1対多 (1対n)



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

モデル

App/Models/Area.php

// 親->子 複数形

```
public function shops()
{
    return $this->hasMany(Shop::class);
}
```

App/Models/Shop.php

```
public function area()
{
    return $this->belongsTo(Area::class);
}
```

ルートとコントローラ

`routes/web.php`

```
use App\Http\Controllers\ShopController;  
Route::get('shops', [ ShopController::class,  
    'index'])
```

コントローラ

```
php artisan make:controller ShopController
```

コントローラ

ShopController.php

```
use App\Models\Area; use App\Models\Shop;
```


```
public function index()
{
    // 1対多 主->従
    $shops = Area::find(1)->shops;
    // 主 <- 従
    $area = Shop::find(3)->area->name;
    dd($area, $shop);
}
```


外部キー制約

親側で登録しているid以外は登録できなくする(制約をかける)場合

create_shops_table.php (子)

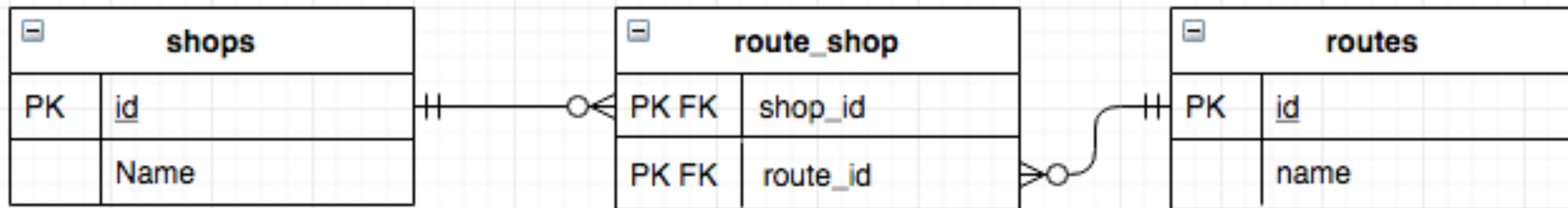
```
Schema::create('shops', function (Blueprint $table) {  
    $table->id();  
    $table->string('name', 20);  
    $table->foreignId('area_id')->constrained();  
    $table->timestamps();  
});
```



リレーシヨン (多対多)

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 メソッド

必要ファイル作成

マイグレーション

テーブル名はアルファベット順で書く(どちらも単数系)

```
php artisan make:migration
```

```
create_route_shop_table
```

シーダー(ダミーデータ)

```
php artisan make:seeder RouteShopSeeder
```

モデルにリレーション

App/Models/Shop.php

```
public function routes()  
{  
    return $this->belongsToMany(Route::class);  
}
```

App/Models/Route.php

```
public function shops()  
{  
    return $this->belongsToMany(Shop::class);  
}
```

マイグレーション

create_routes_table.php

```
Schema::create('areas', function (Blueprint $table) {  
    $table->id();  
    $table->string('name', 20);  
    $table->integer('sort_no');  
    $table->timestamps();  
});
```

create_route_shop_table.php

```
Schema::create('route_shop', function (Blueprint $table) {  
    $table->foreignId('route_id');  
    $table->foreignId('shop_id');  
    $table->primary(['route_id', 'shop_id']); // テーブルには必ずプライマリキーが必要、idを削除する場合は別で指定。複数キーも設定可能  
});
```


RouteSeeder.php

```
use Illuminate\Support\Facades\DB;

public function run()
{
    DB::table('routes')->insert([
        [ 'name' => '山手線', 'sort_no' => 1 ],
        [ 'name' => '京浜東北線', 'sort_no' => 2 ],
        [ 'name' => '東武東上線', 'sort_no' => 3 ],
    ]);
}
```

RouteShopSeeder.php

```
use Illuminate\Support\Facades\DB;

public function run()
{
    DB::table('route_shop')->insert([
        [ 'route_id' => 1, 'shop_id' => 1 ],
        [ 'route_id' => 1, 'shop_id' => 2 ],
        [ 'route_id' => 2, 'shop_id' => 1 ],
    ]);
}
```

DatabaseSeeder.php

```
$this->call([  
    UserSeeder::class,  
    AreaSeeder::class,  
    ShopSeeder::class,  
    RouteSeeder::class, // 親から  
    RouteShopSeeder::class,  
]);
```

ダミー反映は `php artisan migrate:fresh --seed`

ShopController

```
public function index()  
{  
    // 多对多  
    $routes = Shop::find(1)->routes()->get();  
    dd($routes);  
}
```



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

Gitも別講座をご参照ください

gitインストール・gitHubアカウント作成

SSH設定後

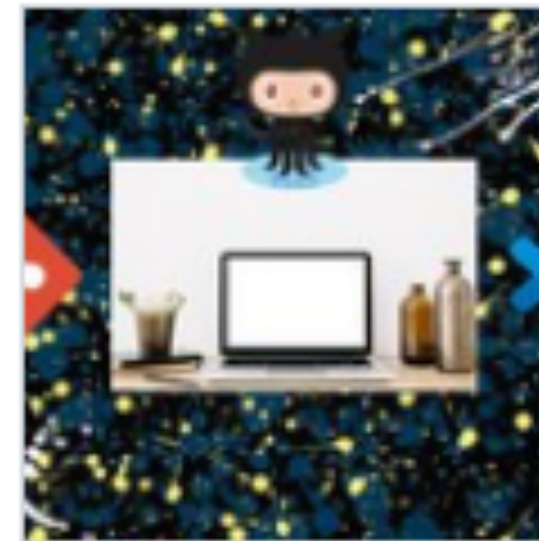
git init

git add -A

git commit -m "first"

git remote origin

git push -u origin main



【Git/GitHub】【初心者向け】最短で実践力を身につけよう【VSCode】イメージ図たっぷり【わ...

ライブ

GitHubからダウンロードする

```
git clone https://github.com/aokitashipro/udemy\_laravel\_basic.git
```

もしくはzipファイルをダウンロード->解凍

```
composer install か composer update
```

```
npm i && npm run dev
```

.env.exampleを.envにコピーし必要情報を記入

```
php artisan migrate:fresh --seed
```

```
php artisan key:generate // キー生成
```

```
php artisan serve
```

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

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)