



Laravel9 その2



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 -> Route -> Controller -> View

モデル・マイグレーション

-m でマイグレーションも同時作成

モデルは単数系 マイグレーションファイルは
複数形で生成される

```
php artisan make:model ContactForm -m
```

xxx_create_contact_forms.php

```
public function up()
{
    Schema::create('contact_forms', function (Blueprint $table) {
        $table->id();
        $table->string('name', 20); // 氏名
        $table->string('email', 255); // メールアドレス
        $table->longText('url')->nullable(); // url null可
        $table->boolean('gender'); // 性別
        $table->tinyInteger('age'); // 年齢
        $table->string('contact', 200); // お問い合わせ内容
        $table->timestamps(); }); }
```

モデル・マイグレーション

php artisan migrate でテーブル生成

よく使うカラム修飾子

nullable(), unique(), unsigned(), after()

マイグレーションは履歴管理

テーブルの履歴を管理できる仕組み。後から列を追加・削除なども履歴を残せる

```
php artisan make:migration add_title_to_contact_forms_table
```

```
public function up() // 追加
```

```
{ Schema::table('contact_forms', function (Blueprint $table) {  
    $table->string('title', 50)->after('name'); }); }
```

```
public function down() // ロールバック
```

```
{
```

```
    Schema::table('contact_forms', function (Blueprint $table) {
```

```
        $table->dropColumn('title');
```

```
    }); }
```

マイグレーションは履歴管理

php artisan migrate // マイグレーション実施

php artisan migrate:status // 状態表示

php artisan migrate:rollback // 1つ戻す

php artisan migrate:rollback --step=2 // 2つ戻す

php artisan migrate:refresh // ロールバックして再実行

php artisan migrate:fresh // テーブル削除して再実行

以前書いていたコードの復元

// LaravelBreezeインストールした事でrouteファイルが書きかわっている

routes/web.php

```
use App\Http\Controllers\TestController.php
```

```
Route::get('tests/test', [ TestController::class, 'index' ]);
```

migrate:freshするとDB内データが全て削除されるので
事前にダミーデータを作っておくのが一般的

リソースコントローラー

よく使うメソッドをまとめて生成できる

```
php artisan make:controller ContactFormController --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

ルート側をまとめてかくなら

```
routes/web.php
```

```
use App\Http\Controllers\ContactFormController.php
```

```
Route::resource('contacts', ContactFormController::class)
```

```
->middleware(['auth']);
```

```
// ルート名もまとめて作成される
```

php artisan route:listで7つのルートが追加されているのが
確認できます

今回は練習も兼ねて別の方法で進めていきます

routes/web.php

```
use App\Http\Controllers\ContactFormController.php
```

```
// 1行ずつ書くならこうなる
```

```
Route::get('contacts', [ ContactFormController::class, 'index'])->name('contacts.index');
```

```
// グループ化してまとめるとシンプルに書ける
```

```
Route::prefix('contacts') // 頭に contacts をつける
```

```
    ->middleware(['auth']) // 認証
```


```
    ->name('contacts.') // ルート名
```

```
    ->controller(ContactFormController::class) // コントローラ指定(laravel9から)
```

```
    ->group(function(){ // グループ化
```

```
        Route::get('/', 'index')->name('index'); // 名前つきルート
```

```
});
```

Viewファイルを
読み込んでみる

auth/login.blade.php

<x-guest-layout>

頭にx-とつくのはBladeコンポーネント(部品)

クラスをつかうパターン

app/View/Components/配下

使わないパターン

resources/views/components/配下

先にresources側を見て、なかったらクラス側も見てみる

スロット

ヘッダー・フッターなど 共通の箇所をまとめたり
一部だけ他の表示に差し替えたりできる機能

layouts/guest.blade.php

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

```
<x-auth-card>
```

```
  <x-slot name="logo"> 名前付きスロット
```

```
</x-auth-card>
```

x-auth-card.blade.php

```
{{ $logo }} . . 名前付きスロット
```

layouts/guest.blade.php

```
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}"> // {{ }} の中にはPHPが書ける
<head>
    <meta name="csrf-token" content="{{ csrf_token() }}"> // csrf使えるように
    <title>{{ config('app.name', 'Laravel') }}</title> // configヘルパ関数
    @vite(['resources/css/app.css', 'resources/js/app.js']) // cssとjsを読み込み
</head>
<body>
    {{ $header }} ・ ・ 名前付きスロット
    {{ $slot }} ・ ・ スロット
</body>
```


auth/login.blade.php

// route(ルート名)

<form method="post"

action="{{ route('login') }}">

@csrf ・ ・ POSTの時は必須

Tailwindcss

CSSはTailwindcss

<https://tailwindcss.com/>

より深く知りたい場合は別の講座も参照ください



【tailwindcss】CSSが苦手な人向けじっくり取り組んでみよう【次世代のcss】 【わかりやすさ重...

ライブ ¥4,900 - 公開

layouts/app と navigation

layouts/app.blade.php

@include('layouts.navigation') // ファイル読み込み

layouts/navigation.blade.php

{{ Auth::user()->name }} // ユーザー名表示

ブラウザの幅を狭めるとレスポンシブ用の表示に変わる
(ハンバーガーアイコンをクリックでメニュー表示)



Index, Create

新規作成

Indexとcreateをつくってみる

dashboard.blade.phpをコピーしてcontacts/index.blade.phpを作成

ルート->コントローラ->ビューの流れ

Routes/web.php

Route::略

```
->group(function(){  
    Route::get('/', 'index')->name('index');  
    Route::get('/create', 'create')->name('create'); //追記  
});
```

Create



App/Http/Controllers/ContactFormController.php

```
public function create()  
{  
    return view('contacts.create');  
}
```

resources/views/contacts/create.blade.php

フォームを作っていく

CSSはTailBlocksを活用



CSS反映させるには、
npm run dev も同時に実行しておく

フォームレイアウトはTailBlocksを参照

Contactの3つめ

<https://tailblocks.cc/>

Create.blade.php

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

```
@csrf
```

```
// 氏名、件名、メールアドレス、ホームページ、性別、年齢、お問い合わせ内容、注意事項に同意する
```

```
<select name="age">
```

```
<option value="">選択してください</option>
```

```
<option value="1">～19歳</option>
```

```
<option value="2">20歳～29歳</option>
```

```
<option value="3">30歳～39歳</option>
```

```
<option value="4">40歳～49歳</option>
```

```
<option value="5">50歳～59歳</option>
```

```
<option value="6">60歳～</option>
```

```
</select>
```

```
</form>
```




Store 保存処理

Store保存処理

ルート->コントローラ->ビュー

routes/web.php

Route::略

```
->group(function(){  
    Route::get('/', 'index')->name('index');  
    Route::get('/create', 'create')->name('create');  
    Route::post('/', 'store')->name('store'); // 追記  
});
```


コントローラ

ContactFormController.php

フォーム登録・・・PHPでは\$_POSTなど。Laravelでは Request クラス
引数に(Request \$request)

DI (Dependency Injection 依存性の注入)
(メソッドインジェクションとも呼ばれる)

Requestをインスタンス化したものが入ってくる

Requestクラス自体は vendor/laravel/framework/src/Illuminate/Http/Request.php

```
public function store(Request $request)
{
    dd($request, $request->name);
}
```

DBへの保存方法

App/Http/Controllers/Auth/RegisteredUserController.php
を参照

```
モデル名::create([  
    'name' => $request->name,  
    略  
]);
```

createでまとめて登録するにはモデルに\$fillableの記載が必要

App/Models/User.php

```
protected $fillable = [ 'name', 'email', 'password' ];
```


モデルに追記

//フォームのname属性を指定する

App/Models/ContactForm.php

```
protected $fillable = [  
    'name',  
    'title',  
    'email',  
    'url',  
    'gender',  
    'age',  
    'contact',  
    'caution'  
];
```

コントローラ

```
public function store(Request $request)
{ // createメソッドでまとめて登録できる
    ContactForm::create([
        'name' => $request->name,
        'title' => $request->title,
        'email' => $request->email,
        'url' => $request->url,
        'gender' => $request->gender,
        'age' => $request->age,
        'contact' => $request->contact,
    ]);
    return to_route('contacts.index'); // to_route でリダイレクト (Laravel9からの書き方)
}
```




Index 一覽表示

Index コントローラ・ビュー

データベースから情報取得

ContactformController.php

```
public function index()
{
    $contacts = ContactForm::select('id', 'name', 'title', 'created_at')
        ->get();
    return view('contacts.index', compact('contacts'));
}
```

resources/views/contacts/index.blade.php

```
@foreach($contacts as $contact)
    {{ $contact->id }} {{ $contact->name }} {{ $contact->title }} {{ $contact->created_at }}
@endforeach
```


Index ビューレイアウト



テーブルのレイアウトは

TailBlocksのPricing 2つめを参考に

<https://tailblocks.cc/>

ナビゲーション



layouts/navigation.blade.php

2つのコンポーネントをコピーして

必要な箇所を修正する

```
<x-nav-link :href="route('contacts.index')">
```

```
<x-responsive-nav-link :href="route('contacts.index')">
```



Show 詳細表示

Show ルート・コントローラ

ルート->コントローラ->ビュー

routes/web.php

略->group(function(){

Route::get('/{id}', 'show')->name('show'); // {id}でviewからのルートパラメータを取得できる
});

ContactFormController.php

public function show(\$id) // 引数にid

{

\$contact = ContactForm::find(\$id); // 1件だけ取得

return view('contacts.show', compact('contact'));

}

Show ビュー

ルート->コントローラ->ビュー

resources/views/contacts/show.blade.php

{{ \$contact->id }} // 1件なのでforeach

は不要

// routeの第2引数でルートパラメータを渡せる

resources/views/contacts/index.blade.php

 \$contact->id]) }}">詳細を見る

性別・年齢の表示を変える

ContactFormController.php

```
public function show($id)
```

```
{
```

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

```
    if($contact->gender === 0)
```

```
    { $gender = '男性'; } else { $gender = '女性'; }
```

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

```
    if($contact->age === 2 ){ $age = '20歳~29歳'; }
```

```
    if($contact->age === 3 ){ $age = '30歳~39歳'; }
```

```
    if($contact->age === 4 ){ $age = '40歳~49歳'; }
```

```
    if($contact->age === 5 ){ $age = '50歳~59歳'; }
```

```
    if($contact->age === 6 ){ $age = '60歳~'; }
```

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

```
}
```




Edit 編集機能

Edit ルート・コントローラ

ルート->コントローラ->ビュー

routes/web.php

```
略->group(function(){  
    Route::get('/{id}/edit', 'edit')->name('edit');  
});
```

ContactFormController.php

```
public function edit($id) // 引数にid  
{  
    $contact = ContactForm::find($id);  
    return view('contacts.edit', compact('contact'));  
}
```


Edit ビュー

ルート->コントローラ->ビュー

resources/views/contacts/edit.blade.php

create.blade.phpをコピー

value="{{ \$contact->name }}" などを追記

resources/views/contacts/show.blade.php

```
<form method="get" action="{{ route('contacts.edit', ['id' =>
$contact->id]) }}">
</form>
```


Edit ビュー

ルート->コントローラ->ビュー

resources/views/contacts/edit.blade.php

年齢と性別はタグの中に@ifで判定を入れる

gender

```
<input @if($contact->age === 0 ) checked @endif>
```

age

```
<option @if($contact->age === 1 ) selected @endif>
```



update 更新

Update ルート・コントローラ

routes/web.php

```
Route::post('/{id}', 'update')->name('update');
```

ContactFormController.php

```
public function update(Request $request, $id)
{
    $contact = ContactForm::find($id);
    $contact->name = $request->name;
    $contact->title = $request->title;
    $contact->email = $request->email;
    $contact->url = $request->url;
    $contact->gender = $request->gender;
    $contact->age = $request->age;
    $contact->contact = $request->contact;
    $contact->save();
    return to_route('contacts.index'); }
```


Update リンク

edit.blade.php

```
<form method="post"  
action="{{ route('contacts.update', ['id'  
=> $contact->id ])}}">  
</form>
```



destroy 削除

Destroy ルート・コントローラ

// formタグの場合はgetかpostしか使えないのでpostでURLを少し変更

routes/web.php

```
Route::post('/{id}/destroy', 'destroy')->name('destroy');
```

ContactFormController.php

```
public function destroy($id)
{
    $contact = ContactForm::find($id);
    $contact->delete(); // deleteで削除

    return to_route('contacts.index');
}
```


Destroy Show.blade.php

クリックしてすぐに消えると誤操作のリスクもあるので
消してもいいかという確認メッセージをJavaScriptで表示
JSで操作するためにidを追記

Show.blade.php

```
<form method="post" action="{{ route('contacts.destroy', [ 'id' =>  
$contact->id ])) }}" id="delete_{{ $contact->id }}">  
@csrf  
<a href="#" data-id="{{ $contact->id }}" onclick="deletePost(this)">削除  
する</a>
```

Destroy JSで対応

Show.blade.php

引数 e でフォーム内の情報を取得

e.dataset.id で data-id で設定した情報を取得できる

<!-- 確認メッセージ -->

<script>

function deletePost(e){

 'use strict'

 if(confirm('本当に削除していいですか?')){

 document.getElementById('delete_' + e.dataset.id).submit()

 }

}

</script>