ミドルウェア

1.ログインしていないと機能を使えない様にするには。

デフォルトのままだとログインしていなくても url から直接飛ぶことでログインしていなくてもマイページなどに飛べてしまう。
一応アクション内に『ログインしているかどうか』の確認処理を挟む事で
飛ばない様にできるが冗長(じょうちょう)になってしまう。
そんなときは Laravel にあるミドルウェア機能を使うと良い。

☆ミドルウェア機能とは
laravel にはアクション実行前や実行後に何らかの処理を実行させたい時に
使う機能。
fuelphp で言う所の before_action と同じ

ミドルウェアはアクション(エンドポイント)ごとに設定ができるローカルミドルウェアとグローバルミドルウェアがある。

☆ミドルウェアの作成
ローカルミドルウェアでもグローバルミドルウェアでも作る物は同じ。
最初は

php artisan make:middleware (アクションに基づいた名前を書く)

今回は
php artisan make:middleware CheckLoggedIn

例

```
<?php

namespace App\Http\Middleware;

use Closure;

class CheckloggedIn
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
```

```
    public function handle($request, Closure $next)
    {

        //Auth::check()・・・ユーザーがログインしているかをtrueかfalseで返す処理
        if(!Auth::check()){
            //ログインしていなかった場合ルーティング内のname(‘login’)を通してログインページに返す処理を走らせる。
            return redirect(‘login’);
        }
        //指定アクションを後から走らせる処理
        return $next($request);
    }
}
```

アクション実行後にミドルウェア機能を走らせたい場合

```
 public function handle($request, Closure $next)
    {
        //指定アクションを先に走らせる処理
        $response = $next();

        if(!Auth::check())
        {
            //ログインしていなかった場合ルーティング内のname(‘login’)を通してログインページに返す処理を走らせる。
            return redirect(‘login’);
        }
    }
```

グローバルミドルウェアの設定の仕方

あるミドルウェアをアプリケーションの全HTTPリクエストで実行したい場合は
app\Http\Kernel.phpクラスの$middlewareプロパティへ追加します。

ローカルミドルウェアの設定の仕方
特定のルート(アクション)のみに対しミドルウェアを指定したい場合はまず
app\Http\Kernel.phpファイルにミドルウェアの短縮キーを登録する。

Kernel.phpの53行目あたりの↓に

```
protected $routeMiddleware = [
        'auth' => \App\Http\Middleware\Authenticate::class,
```

```
        'auth.basic' => \Illuminate\Auth\Middleware\
AuthenticateWithBasicAuth::class,
        'bindings' => \Illuminate\Routing\Middleware\
SubstituteBindings::class,
        'cache.headers' => \Illuminate\Http\Middleware\
SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
        'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
        'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' => \Illuminate\Auth\Middleware\
EnsureEmailIsVerified::class,

        //ミドルウェア機能の処理内容をなるべくわかる様に名前は書く。短縮キーは
ミドルウェアファイルの pass+語尾に::class を書く。
        'check' => \App\Http\Middleware\CheckloggedIn::class,
    ];
```

設定後にルーティングファイルに->middleware(設定名)を追加してミドルウェア機能を走らせる。
例

```
Route::get('/drills','DrillsController@index')->name('drills')-
>middleware('check');
```

デフォルトで作成されている物も使えるので後で見る。

一つ一つに設定するのが面倒くさい場合 Route::group を使うと良い。
例

```
Route::group(['middleware' => 'check'],function(){
  Route::post('/drills','DrillsController@create')->name('drills.create');
  Route::get('/drills/new','DrillsController@new')->name('drills.new');
  Route::get('/drills','DrillsController@index')->name('drills');
  Route::get('/drills/{id}/edit','DrillsController@edit')->name('drills.edit');
  Route::post('/drills/{id}','DrillsController@update')-
>name('drills.update');
  Route::post('/drills/{id}/delete','DrillsController@destroy')-
>name('drills.delete');
  Route::get('/drills/{id}','DrillsController@show')->name('drills.show');
});
```