ビューコンポーザー(laravel)
ビューで表示させるために通常の MVC の FW では、そのロジックをビューやコントローラー(主にコントローラー)に書いていましたが、そうすると処理が多くなる程読みにくいコードになっていく。

ビューにはほとんどロジックは書かずにただ『変数の中身を表示するだけ』にした方がコードが読みやすい。

・ DB へ接続して CRUD 処理
・ ビューへ渡す値を作る
・ その値をビューへ渡す

というためのコードを毎回書いていたが
毎回お決まりのコードだと書かずにしたい所。
Laravel では、こういった悩みを解決してくれるのが『ビューコンポーザー』という機能です。

例えば、下記のようなブログの一覧と詳細画面があったとするとそこには『サイドバー』が大抵あり、どちらのアクションでも同じ情報を取得して表示する処理を書かなければいけません。

```
<?php

class BlogController exends Contoller
{
  public function index()
  {
     //メインコンテンツ
     $posts = Post::all();

     //人気記事
    $ranking_posts = Post::lanking();

    //最近の投稿
    $latest_posts = Post::latest(5);

    //カテゴリ
    $categories = Category::all();
    foreach($categories as $key => $category){
        $category->count = category::where('id',$category->id)->count
        $categories->put($key,$category);
    }
  }
}
```

簡単にまとめれば『お決まりのコントローラー・ビューの処理は別ファイルにまと

めておこう』という感じ。

下の様な毎回ログインメンバーの名前を表示する処理などは基本毎回使うものなので別にまとめておく。

```
$user = \Auth::user();
if($user)  {
echo “hello $user->name”;
}
```

実際に作る

その一

ビューコンポーザーの作り方

ビューコンポーザーの作り方はざっくりと 3STEP あり
1.サービスプロバイダクラスの作成
2.ビューコンポーザークラスの作成と compose メソッドの実装
3.サービスプロバイダクラスの boot メソッドでビューコンポーザークラスを作成する

になる。

1.サービスプロバイダクラスの作成

サービスプロバイダクラスの作成
serviceprovider クラスは artisan コマンドで簡単に作れる。
サービスプロバイダというのは、『アプリケーションの準備段階で何かしらの処理や設定を行うためのクラスをして用意されている物です。

名前は基本何でも大丈夫だけどコンポーザーを提供するので名前は
『ComposerServiceProvider』にした方が良い。
(ていうかだいたいこれっぽい)

コマンド例
php artisan make:provider ComposerServiceProvider

で http\provider\の配下に出来上がる。

2.ビューコンポーザークラスの作成と compose メソッドの実装

ビューコンポーザークラスは自分の手で作る。
ファイルは app\http 下に作った ViewComposers

作成例(UserComposer)
*継承は無し。

```
<?php

namespace App\Http\ViewComposers;

use Illuminate\Contacts\View\View;

class UserComposer{

      public function compose(view $view)
      {
            $view->(‘user’,’ユーザー情報を第二引数へ’);
      }
}
```

今回の場合だと

```
<?php

namespace App\Http\ViewComposers;

use Illuminate\Contacts\View\View;
use Illuminate\Contacts\Auth\Guard;

class UserComposer{
      protected $auth;

      public function compose(Guard $auth)
      {
          $this->auth = $auth;
      }

      public function compose(View $view)
      {
          // userという変数を使える様にし、その中に$this->auth->user()という
値を詰める。
          $view->with(‘user’,$this->auth->user());
      }
}
```

==========

```
<?php
```

```
namespace App\Providers;

use Illuminate\Support\ServiceProvider;

class ComposerServiceProvider extends ServiceProvider
{
    /**
     * Register services.
     *
     * @return void
     */
    public function register()
    {
        //
    }

    /**
     * Bootstrap services.
     *
     * @return void
     */
    public function boot()
    {
        //連想配列で流します
        //キーにコンポーザを指定し、値にビューを指定します(ワイルドカードも使えます。)
       //ワイルドカード･･･ 何かしら情報を引っ張ってくる処理等に使う条件式内などに使う
       文字が入る時などに使う。
        piyo*.txt･･･ 中に 0 文字以上の何かが入る。
        piyo?.txt･･･ 中に 1 文字入る。
        //この場合,layouts ディレクトリ配下のビューテンプレートが読み込まれた場合に UserComposer を読み込み(=Super が作られる)という設定の仕方になります。
        View::composer([
            UserComposer::Class => 'layouts.*'
        ])
    }
}
```

==========

11.5 config/app.php に設定を追加

config/app.php の providers に ComposerServiceProvider を追加する事で ComposerServiceProvider を laravel が自動で実行できる様になる。

```
‘providers’ => [
ComposerserviceProvider::class,
```

],

ビューコンポーザーの作成

1.サービスプロバイダクラスの作成
2.ビューコンポーザクラスの作成と compose メソッドの実装
3.ビューコンポーザークラスの boot メソッドでビューコンポーザクラスを利用する

1.コマンド打つ
php artisan make:provider ComposerServiceProvider

2.