## Create new project

$ composer create-project laravel/laravel projectName "5.0.\*" --prefer-dist

$ composer update

## Start server

$ php artisan serve

## Update layout file

remove google font

resources/views/app.blade.php

## Update DB details in .env

DB\_HOST=localhost

DB\_DATABASE=laravel

DB\_USERNAME=aleverett

DB\_PASSWORD=stroses

## Update Mail details and queue drive in .env

MAIL\_DRIVER=smtp

MAIL\_HOST=smtp.livemail.co.uk

MAIL\_PORT=25

MAIL\_USERNAME=alexe@wts-group.com

MAIL\_PASSWORD=roses111!

QUEUE\_DRIVER=database

## in config/mail.php

'from' => ['address' => 'alexe@wts-group.com', 'name' => 'Alex Everett'],

'encryption' => '',

## Create views for static pages

resources/views/pages

create empty files in pages for static pages index.blade.php, about.blade.php, contact.blade.php

## Typical blade file

@extends('app')

@section('title', '- Home Page')

@section('content')

<div class="container">

<h1>Home</h1>

</div>

@endsection

## Create static pages controller

App/Http/Controllers/PagesController for all static pages, index, contact, about etc

$ php artisan make:controller PagesController --plain

Add methods in controller eg

public function index()

{ return view('pages.index');

## Add routes for static pages

app/Http/routes.php eg.

Route::get('/', 'PagesController@index');

Route::get('/contact', 'PagesController@contact');

## Get form and html helpers

Begin by installing this package through Composer. Edit your project's composer.json file to require laravelcollective/html

"require": {

"laravelcollective/html": "~5.0"

}

Next, update Composer from the Terminal:

$ composer update

Next, add your new provider to the providers array of config/app.php:

'providers' => [

'Collective\Html\HtmlServiceProvider',

],

'aliases' => [

'Form' => 'Collective\Html\FormFacade',

'Html' => 'Collective\Html\HtmlFacade',

],

## Create contact form

make named route for the contact form post. Use named routes for forms just in case the url changes, you can keep the name of the route

Route::post('contact', [ 'as' => 'post.contact', 'uses' => 'PagesController@postContact' ] );

resources/views/pages/contact.blade.php

open the form

{!! Form::open(array('route' => 'post\_contact', 'class' => 'form-horizontal')) !!}

Add form fields

<div class="form-group">

{!! Form::label('name', 'Your Name', ['class' => 'col-md-4 control-label']) !!}

<div class="col-md-6">

{!! Form::input('text', 'name', null, ['class' => 'form-control']) !!}

</div>

</div>

'text' is the input type, 'name' is the input name, null is the input default value, then an array of any other attributes

Close the form

{!! Form::close() !!}

## Create method in PagesController to handle the form post

public function postContact(Request $request) {}

$data = $request->all();

Mail::send('emails.enquiry', $data, function($message)

{

$message->to('aleverett50@hotmail.com', 'Alex Everett')->subject('Website Enquiry');

});

Add redirect back to contact form after script

return redirect('contact')->withSuccess('Your message has been sent!'); or

return redirect('contact')->with('success', 'Your message has been sent!');

## On view page somewhere near <h1> tag

@if ( Session::has('success') )

<div class="alert alert-success">

{{ Session::get('success') }}

</div>

@endif

## Create the html that will email the enquiry to the site owner

Resources/views/emails/enquiry.blade.php

@extends('emails.app')

@section('content')

test email {{ $name }}

$name is one of the values of the $data array on the Mail script on the PagesController

@endsection

Create a layout template in resources/views/email/app.blade this will have the header and footer of every email that gets sent out.

## Run emails from a database queue

Change default to database in config/queue.php

$ php artisan queue:table then $ php artisan migrate this will create a table ‘jobs’ to hold all the queues.

## Make migrations files for DB tables

add current migration files in to database (users and password\_resets)

Change users table to a relevant table, eg. first\_name, last\_name etc

$ php artisan migrate (adds the tables)

To reverse a migration

$ php artisan migrate:rollback

to create a new table migration file

$ php artisan make:migration create\_posts\_table --create=posts

this creates a migration file which would create a table posts. you just need to add the fields, id and timestamps already added.

to make an update to an existing table

$ php artisan make:migration add\_status\_to\_users\_table --table=users

to add a new field just put the field

$table->string('status')->after('remember\_token')->nullable();

with

$table->dropColumn('status');

to remove that

to update an existing field

$table->string('first\_name', 50)->change();

this would change the first\_name field to have 50 chars.

would need to have the reverse in down()

$table->string('first\_name', 255)->change();

## Seed tables with data

to seed the database, put dummy rows in

database/seeds

create a file/class UserTableSeeder.php

add

use Illuminate\Database\Seeder;

use App\User;

at top of each seed class

make method run(){}

delete all rows

DB::table('users')->delete();

add row data

User::create([ 'email' => 'alexe@wts-group.com', 'first\_name' => 'AL', 'last\_name' => 'Everett', 'password' => Hash::make('pass') ]);

Then in DatabaseSeeder.php uncomment

$this->call('UserTableSeeder'); UserTableSeeder must match the class name it is seeding

$ php artisan db:seed

or $ php artisan db:seed --class=UserTableSeeder for single table seed

if it doesn't work run $ composer dump-autoload

## From register fields will need changing (first name, last name)

app\Services\Registrar to change register fields

## Login redirect urls can be changed

Remove vendor/compiled.php otherwise methods in there may override real classes

class AuthController extends Controller {

protected $redirectAfterLogout = '/auth/login';

protected $redirectTo = '/'; go after logging in

to get auth/ out of url do in routes.php

Route::controllers([

'/' => 'Auth\AuthController',

'password' => 'Auth\PasswordController',

]);