Content

What is Laravel?

- Love beautiful code? We do too.
- The PHP Framework.
- Authorization.
- Object-Oriented Libraries.
- Artisan.
- MVC Support.
- Security.
- Database Migration.
- Great Tutorials (Laracasts).
- Blade Templating Engine.
- Automatic Package Discovery.
- https://laravel.com/docs/5.8
- https://laracasts.com

Composer

- Composer is a dependency manager
- Download From here

  https://getcomposer.org/download/
- Make composer globally

  https://getcomposer.org/doc/00-intro.md#globally

Install Laravel

- Via Composer Create-Project

  composer create-project --prefer-dist laravel/laravel blog

- https://laravel.com/docs/5.8#installing-laravel

Directory Structure

- app/ --> where your application go like (models controllers)
- bootstrap/ --> files laravel uses to boot every time
- config/ --> configuration files like (database configuration)
- database/ --> where (migrations - seeds - factories) exist
- public/ --> directory where server points to it when serving
- request it also contains (index.php)
- resources/ --> contains (views - Saas -Less files - ..)
- routes/ --> all routes definitions (Api - console -http)

Directory Structure continue

- storage/ --> where cache logs compiled files exist
- tests/ --> where unit & integration test exist
- vendor/ --> composer packages
- .env --> defines environments variables
- .env.example --> same as above ,it should be copied when
- cloning other projects
- .gitattributes , .gitignore --> git configuration files
- artisan --> php script to run artisan commands

Directory Structure continue

- composer.json , composer.lock --> contains project
- dependencies
- package.json --> like composer.json for frontend assets
- phpunit.xml --> configuration for php unit
- readme.md --> markdown for laravel introduction
- server.php --> it’s a server that emulate apache
- mod_rewrite
- webpack.mix.js --> used for compiling and mixing frontend
- assets

Routing

- Controller’s Method

  Route::get(‘/posts’ , ‘PostsController@index’);

- Parameters

  Route::get(‘/posts/{id}’ , ‘PostsController@edit’);

Routing continue

- Named Routes

  Route::get(‘/posts’ , [

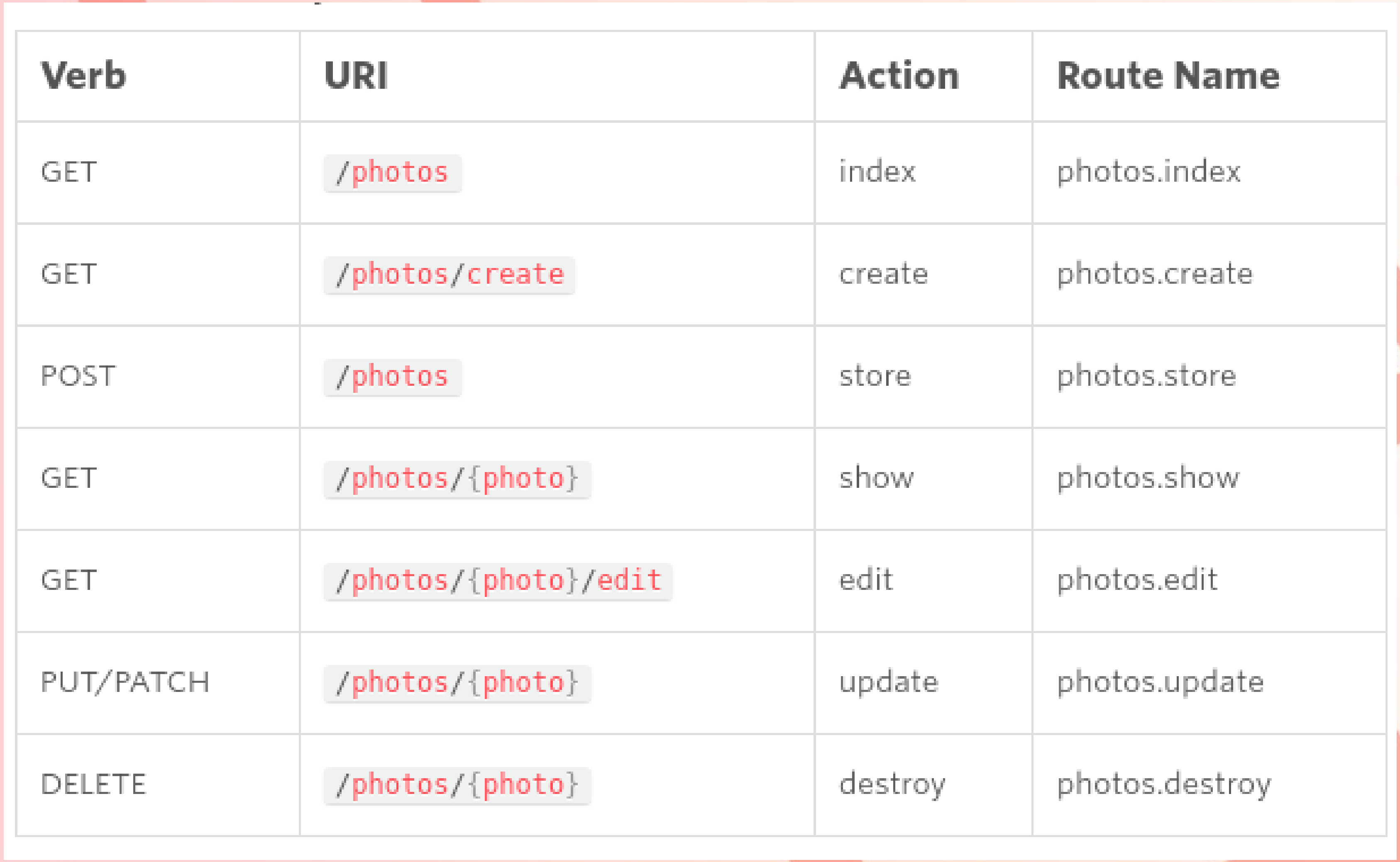
  ‘uses’ => ‘PostsController@index’,

  ‘as’ => ‘posts.index’

  ]);
- Resource Controllers

  Route::resource('posts', '
  PostsController');

Actions Handled By Resource Controller



Verb	URI	Action	Route Name
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

More at :
https://laravel.com/docs/5.8/controllers#resource-controllers
https://laravel.com/docs/5.8/routing

Blade

- echo data :-

  {{ $post }} instead of <?php echo $post ; ?>

- conditions :-

  @if ($post->name == ‘firstPost’)

  // do some stuff

  @endif

- looping :-

  @foreach($posts as $post) //do some stuff @endforeach

- - inheritance :-

  @exnteds(‘layouts.master’)

Blade continue

- Echo data :-

  {{ $post }} instead of <?php echo $post ; ?>
- Conditions :-

  @if ($post->name == ‘firstPost’)

  // do some stuff

  @endif
- looping :-

  @foreach($posts as $post) //do some stuff @endforeach
- Inheritance :-

  @exnteds(‘layouts.master’)

Blade continue

- Define sections :-

  @section(‘content’)

  <h1> hello from the content

  @endsection
- Printing sections :-

  @yield(‘content’)
- Including :-

  @include(‘scripts’)
- Including with parameters :-

  @include(‘post.form’,[‘method’ => ‘POST’])
- More at https://laravel.com/docs/master/blade#introduction

Database & Eloquent

- Laravel ORM is called Eloquent
- Database configuration in .env file or from config/database.php.
- Laravel migration helps in making database persistent across multiple machines .
- DB facade used to form query builder object

Database & Eloquent continue

- $post = DB::table(‘posts’)->find(20);

  //finds a row with id =20
- $posts = DB::table('posts')->get();

  //get all rows in posts table
- $singlePost = DB::table(‘posts’)->where(‘slug’ ,‘FirstPost’)->first();

  //where conditions to query
- $firstPost = DB::table(‘posts’)->first();

  //gets the first row

Database & Eloquent continue

- DB::table(‘posts’)->insert(

  [‘title’ => ‘first post title’ , ‘desc’ => ‘first post desc ‘]

  ); //inserting a row
- DB::table('posts')->where(‘id’ , 1)->delete();

  //deletes a row
- DB::table(‘posts’)->where(‘id’ , 1)

  ->update([‘title’ => ‘changed title post]);

  //update the post title only

Database & Eloquent continue

- php artisan make:model Post //create a new model class
- Laravel by default gets the plural name of model as a table name and makes query based on that
- Post::all() //will search in posts table and get all rows
- Post::create([‘title’ => ‘first post’ , ‘desc’ => ‘desc post’); //this will give a MassAssignmentException unless you override $fillable

Database & Eloquent continue

- Post::find(25)->update([‘title’ => ‘update post title’)

  //updates title for post with id 25
- Post::where(‘votes’, 23)->delete()

  //deletes any post have votes with 23
- To define a relation in Post model you will define a function in

  the class for example i want to say post have many sections .

  //in User model class
- public function posts ()

  { return $this->hasMany(Post::class);}

  // in controller for example

  $posts = User::find(1)->posts;

Database & Eloquent continue

- Remember that query results are collection objects
- More at Eloquent

  https://laravel.com/docs/master/eloquent#introduction
- More at Collections

  https://laravel.com/docs/master/collections#available-methods

Request Validation

- Request Life Cycle :-

  See index.php first it loads the composers's autload file .
- Then we bootstrap laravel application container and register some basic service providers like Log Service provider look at Illuminate/Foundation/Application.php constructor method .
- Finally we create instance of kernel to register providers and take user request and process it through middlewares & handle exception then return responseRequest Validation

Request Validation

- In Controller :-

```
$this->validate( $request , [
‘title ‘ => ‘required ‘,
‘desc’ => ‘required|max:255’
] ,
[
‘title.required’ => ‘title is required to be filled ‘ ,
‘desc.max’ => ‘description max num of chars is 255 ‘
]);
```

Request Validation

- Another way with request file :-

  php artisan make:request PostsStoreRequest

  Then in authorize method make it return true.

  After that define your rules in rules method.

- More at

  https://laravel.com/docs/5.8/validation#available-validation-rules

  https://laravel.com/docs/5.8/validation#customizing-the-error-messages

Laravel Auth

- More at

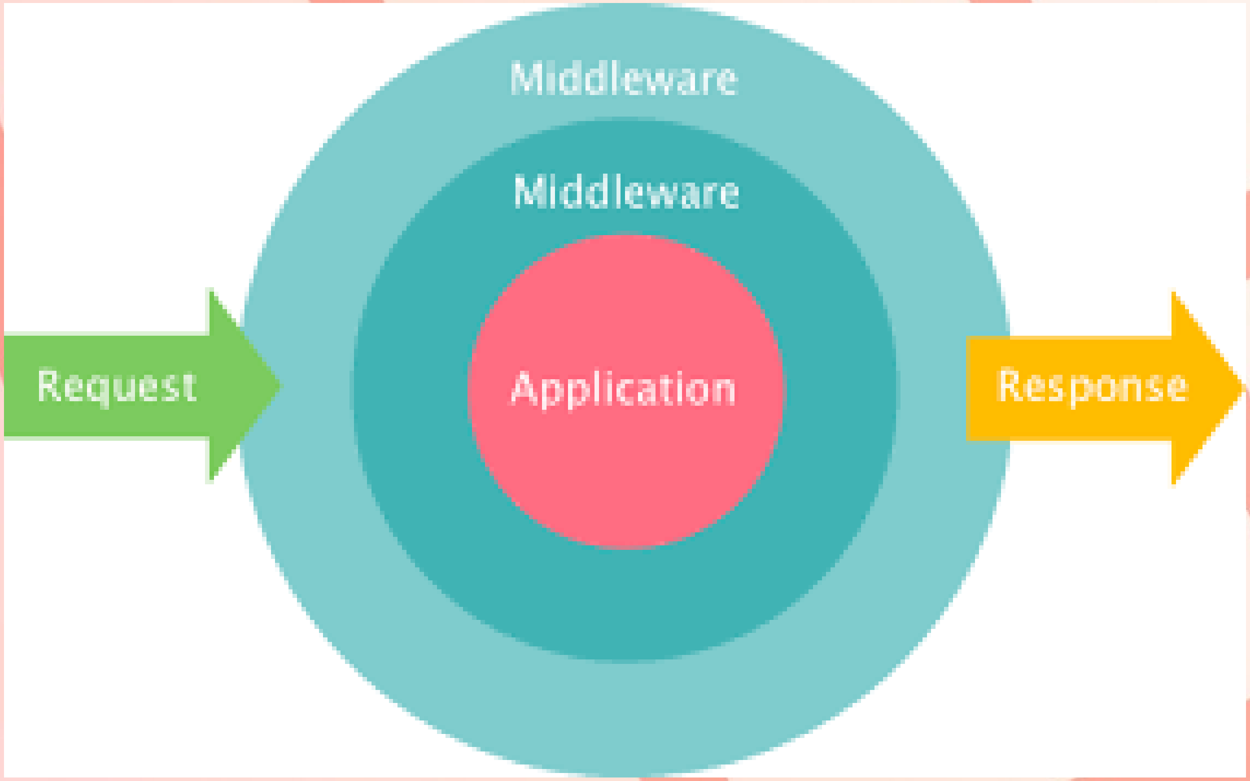
  https://laravel.com/docs/5.8/authentication#authentication-quickstart

  php artisan make:auth

Middleware

- It’s a series of layers around the application , every request passes through middlewares .
- Middlewares can inspect the request to decorate or reject it
- Also middlewares can decorate the response .
- register the middlewares in app/Http/Kernel.php
- handle($request , ..) method where you handle the request and choose to pass it for the next middleware or not .

Middleware



Service Container

- Other names for service container

  (IOC container , DI container , Application container)
- Dependency Injection (DI) :-

  Instead to make object instantiate it's dependencies

  internally , it will be passed from outside.
- Read more at

  https://laravel.com/docs/5.8/container

Happy code

- Our application will be simple Blog system:
- Where you can write a posts
- Comment on a posts
- See all posts and edit posts
- User can login and register
- User can see all posts, edit or delete
- All posts have pagination, a post have a title, content and a photo
- Title not greater that 30 characters
- Content not greater that 255 characters
- The photo, title and content is required.
- Make created at date readable and use soft delete for posts.