# ***------GETTING STARTED------***

**RELEASE**

* Laravel Nova admin Dashboard
* Optional Email Verification (auth scaffholding)
* Support for guest users in authorization gates & policies

**INSTALLATION**

- Install

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

- Basic Configuration

> Entry point = “*public/index.ph*p” > Check *config* folder

> Folder permissions [ *sh-start.sh ]* > .*env* file

**CONFIGURATION**

* Using *DotEnv* php library for **.env** 
  + Any variable in **.env** can be overwritten by *server/system* environment variables
  + **$\_ENV (global php variable)**
  + *‘debug’ => env( “APP\_DEBUG”, false )* (in config files)
  + *$environment = App::environment( )*
* Access config :- *$a = config::( ‘app.timezone’ )*
* Maintenance mode
  + *php artisan down*  ( no queued jobs handled)
  + *php artisan up (* queue resume )
  + View file :- */views/errors/503.blade.php*
  + Envoyer is alternative to maintenance mode with zero-downtime.

**DIRECTORY**

* *app , bootstrap, config, database, public, resources, routes, storage, test, vendor*
* *app*
  + *Broadcast, Events, Jobs, Listeners, Mail, Notifications, Policies, Rules* => (created by *artisan make:xyz* command
  + *Console, Exceptions, Http, Providers* => (Created by default)

**DEPLOYMENT:**

* Nginx configuration
* *composer install --optimize-autoloader --no-dev*
* *php artisan config:cache*
* *php artisan route:cache*

***------ARCHITECTURAL CONCEPTS------***

**REQUEST LIFECYCLE**

* ***public/index.php***
  + require => /*vendor/autoload.php*
  + require once => **$app** as instance of *bootstrap/app.php*
  + **$app** is new instance application which is indeed a ***service-container***
  + Request is sent to ***http/console kernels*** 
    - *app/Http/Kernel.php*
    - *app/Console/Kernel.php*
  + Http Kernel works on
    - Tasks needed before request handling
      * *examples :- logs, error handling, env variables, etc*
    - Middlewares
      * *examples :-* *session, maintenance mode, csrf, etc.*
    - Takes *http request* gives *http response*
    - Load ***service providers***  from *config/app.php*
      * *First call for all providers =>* ***register()***
      * *Then call each providers* => ***boot()***
      * *examples:- database, queues, validation, routing, etc*
    - Dispatch request to *router (route/controller)*
    - Return reponse.

**SERVICE CONTAINER**

* Makes dependency(finding classes) easier by *Dependency Injection.*
* **Binding** 
  + Registering **Class** or **Interface**(*Service providers*) in the **container** is called binding
  + Makes finding dependent classes easier by
    - Using **namespaces** instead of file path
    - Giving **instance** of required (dependent) classes whenever called.
      * new instance [ **bind()** ] => eg:- *User model* instance
      * same instance [ **singleton()** ] => eg:- *Request* instance
  + ***register( )*** inservice provider does binding.
  + No need to *bind* class if it does not depend on any **interfaces**
* Access container in service provider => **$this->app**
* **Simple Binding** [ ***bind() ]***
  + Returns a *new instance object*
  + Example:-
    - *$this->app->****bind****( “HelpSpot\API” , function($app){*

*return new HelpSpot\API($app->make(“HttpClient”));*

*});*

* **Binding a Singleton [ *singleton() ]***
* Resolved only once
* Returns **same** instance **each time**
* Example:-
  + *$this->app->****singleton****(“HelSpot\API”, function( $app){*

*return new HelpSpot\API($app->make(“HttpClient”));*

*});*

* **Binding Instance** [ ***instance() ]***
* Bind existing object instance
* Example :
  + *$obj = new HelpSpot\API( new HttpClient ) ;*

*$this->app->****instance****( “HelpSpot\API”, $obj );*

* **Binding primitives**
* If class requires other class instance and **primitive**(int, string etc) value too
* Example:-
  + *$this->app->****when****(“App\Http\Controller\UserController”)*

*->****needs****(“$anyVariable”)*

*->****give****($value);*

* **Binding Interface To Implementation**
  + Say *=> class RedisEventPusher* ***implements*** *EventPusher*
  + Then if ***EventPusher*** obj is called (type-hint) then give ***RedisEventPusher*** object
  + Example:-
    - $this->app->bind(

“App\Contracts\EventPusher”,

“App\Services\RedisEventPusher”

);

* **Resolving**
  + ***make()***
    - *$api = $this->app->****make****( “HelpSpot\API” );*
  + **resolve()**
    - $*api =* ***resolve****( “HelpSpot\API” );* ( If $this->app is not accessible )
  + ***Automatic Injection*** *( Type-hint )*
    - Example:-
      * *public function \_\_construct (* ***App\Model\Users $users*** *){*

*$this->users = $users;*

*}*

**SERVICE PROVIDER**

* Bootstraped = registering things like *bindings, event listeners, middleware and routes*
* Register service provider in ***config/app.php***
* Writing a new Service provider
  + *php artisan* ***make:provider RiakServiceProvider***
  + Must extend => *Illuminate\Support\ServiceProvider* class
  + 2 basic methods
    - **register( )**
      * Only binding a class into container
    - **boot( )**
      * Called after all providers are registered.
      * Can type-hint dependency if needed
  + Register service provider in ***config/app.php***
* **Deferred services** loaded when required

**FACADES**

* ***STATIC*** interface to classes in service container
* Working
  + *A new facade class extends baseclass Illuminate\Support\Facades\Facade*
  + Base class uses ***\_callStatic()*** magic-method to pass commands to object resolved from container.
  + example:
    - *class Cache* ***extends*** *Facade*

*{*

// No static methods like***get( )***

*protected static function* ***getFacadeAccessor****( ) {*

// return name of service container binding

// Then laravel resolves name to object and runs ***get( )*** on obj

*return* ***‘cache’*** *;*

*}*

* Using any class as a
  + Dependency injection
    - *use App\Contracts\Publisher;*
    - Apply => *public function publish( Publisher $p )*
  + Facade
    - *use* ***Facade****\App\Contracts\Publisher;*
    - Apply => *Publisher::anyMethod()*

**CONTRACTS**

* Set of ***interfaces***
  + ***eg:*** *Illuminate\Contracts\Queue\Queue*
  + Laravel Implements Queue this with variety of drivers

***--------BASICS--------***

**ROUTES**

* Folder **routes** with 4 files **[ web / api / console / channel ]**
* ***App\Providers\RouteServiceProvider***
  + Uses ‘*web’ middleware* to routes/*web,php*
  + Uses ***/api*** prefix to url with ‘*api’ middleware* to *routes/api.php*
* ***Route :: get*** *(“url” , callback);* ***[ get, post , put , patch, delete, options ]***

*::* ***match*** *( [ ‘get’, ‘post’ ] , “url” , callback );*

*::* ***any****( “url” , callback);*

*::* ***except****( [ ‘get’, ‘post’ ] , “url” , callback);*

*::* ***redirect*** *( ‘/here’, ‘/there’ );*

*:: redirect ( ‘/here’, ‘/there’ , statusCodeLike301 );*

*::* ***view*** *( ‘ url’ , ‘viewFile ‘, [ “key” => “value” ] );*

* CSRF Protection (in Forms) [ ***post, put, delete ]*** 
  + ***@csrf***
* *Route Parameters*  ( Note: cannot use ‘-’ for parameter name )
  + *Route :: get ( “user/****{ id }/****photos/{* ***$pid*** *}” , function (* ***$id , $pid*** *) { …..} );*

*:: get ( “user/****{ name? }****” , function (* ***$name = ”Umesh”*** *) { …..} );*

* Regular Expression
  + *Route :: get ( “user/****{ id }****” , function (* ***$id*** *) { …..} ) ->* ***where ( “id” , “[0-9]+” )****;*
  + *Global Route Parameter* (Applys to all routes parameters with same name )
    - In ***RouteServiceProvider***  method ***boot ( )***
      * *Route ::* ***pattern*** *( ‘id’ , ‘[0-9]+’ );*
* *Named Routes*
  + *Route :: get ( “url” , callback ) ->* ***name ( “****myroute****” )****;*
  + Making urls
    - *route ( “myroute” , [ “id” => 232 ] );*
  + *Redirect url*
    - *redirect( ) -> route ( “myroute” , [ ‘id’ => 232 ] );*
* *Route Group*
  + Merged => *Middlewares , where*
  + Appended => *nameprefixes , namespaces, urlprefixes*
  + *Route ::* ***middleware*** *( [ ‘first’ , ‘second’ ] ->* ***group*** *( function ( ) {*

*Route :: get ( ‘/’ , callback ) ;*

*Route :: post ( ‘user/create’ , callback ) ;*

*});*

*::* ***namespace*** *( ‘Admin’ ) ->* ***group*** *( function ( ) { …… } ) ;*

* + Namespace of group will be *=> App\Http\Controllers\Admin*

*::* ***domain*** *( ‘ { username }.myapp.com’ ) ->* ***group*** *( function ( ) {*

*Route :: get ( project/{ id }’ , function ( $username , $id ) { ….} ) ;*

*}) ;*

*::* ***prefix*** *( ‘admin’ ) ->* ***group*** *( function ( ) { …. } );*

*::* ***name*** *( ‘admin’ ) ->* ***group*** *( function ( ) { ….} );*

* *Route-Model binding*
  + *Implicit* 
    - *Route :: get ( “users/{user}” , function(* ***App\Models\User $user*** *){ …. });*

*Note : variable $user and parameter { user }* has same name.

* If want to use *“slug”* instead of *“id”* as default key for users database table

// In *App/Models/User*

public function ***getRouteKeyName*** ( ) { return “slug” ; }

* *Explicit*
  + Same as Implicit but adding In ***RouteServiceProviders*** method ***boot ( )***
    - Route :: **model**( ‘user’ , ‘App\Models\User :: class ) ;
* *Fallback Route (* Always at end of all routes )
  + *Route ::* ***fallback*** *( callback );*
* *Rate Limiting*
  + ***throttle*** *middleware* is to be assigned to route / group
  + Route :: middleware ( ‘auth:api’ , **‘throttle:60,1**’ ) -> group( function ( ) {.....} ) ;
  + throttle:60,1 => 60 request / 1 minute
* Form method spoofing
  + **@method(‘put’)**
* *Accessing current route*
  + *Route ::* ***current( )*** *; [ current ( ) , currentRouteName ( ) , currentRouteAction ( ) ]*

**MIDDLEWARES**

*- Folder =>* ***app/Http/Middleware***

- *Command* => *php artisan make:middleware CheckAge*

***-*** *Register* => ***app/Http/Kernel.php***

- Global Middlewares => ***$middleware***

- Individual route by adding key name => ***$routeMiddleware***

**-** web / api routes group => *$****middlewareGroups***

**-** Sortingmiddleware => ***$middlewarePriority***

*- Run =>* ***handle( )*** in middleware

***-*** *Before request*

*- public function* ***handle*** *( $request , Closure $next )*

*{*

*// do work here*

*return $next ( $request );*

*}*

***-*** *After request*

*- public function* ***handle*** *( $request , Closure $next )*

*{*

*$response = $next ( $request );*

*// do work here*

*return $response ;*

*}*

**CSRF PROTECTION**

* ***CSRF*** *=> Cross-site request forgery* 
  + ***VerifyCsrfToken***Middleware added in ***web***  middleware group.
  + Unauthorized commands by authenticated user
* Laravel automatically generates csrf ***token*** for each active session
* Blade directive in each form => ***@csrf***
* *Exclude some URI*
  + In *VerifyCsrfToken* => **$except**
* *x-csrf-token* ( *Ajax requests* )
  + Add => *<meta name="csrf-token" content="{{ csrf\_token() }}">*
  + **Axios** HTTP Library
    - Default library by laravel
    - csrf is applied to all ajax request in *resources/js/bootstrap.js*
  + ***Jquery***
    - *$.ajaxSetup ( [*

*headers : {*

*‘X-CSRF-TOKEN’ : $(‘meta[ name = “csrf-token” ]’).attr( ‘content’ )*

*}*

*] ) ;*

* *x-xsrf-token*
  + Csrf stored in ***cookie*** having *“XSRF-TOKEN” key*
  + *Angular, Axios, etc* automatically uses this *key* to set the ***x-xsrf-token*** header

**CONTROLLERS**

* *Path* ***: app/Http/Controllers***
* Extends***Base Controller***( optional to extend ) provides methods like
  + *middleware( ), validate( ), dispatch( )*
* *Route :: get ( “uri” , “****Media\PhotoController@index“*** *) ;*
* ***Invokable*** 
  + Only 1 method ***\_\_invoke( )*** in controller
  + *Route :: get ( “url” , “****MyController****” ) ;* (No *@action* )
* ***Middleware***
  + In ***\_\_construct( )*** method we can add middlewares
  + *$this -> middleware( ‘auth’ ) ;*

*$this -> middleware( ‘log’ ) -> only( ‘index’ ) ;*

*$this -> middleware( ‘subscribe’ ) -> except( ‘index’ );*

* ***Resource / CRUD*** 
  + *php artisan make:controller MyController --****resource***
  + *Route::****resource****( “url” , “MyController” ) ->*
    - ***only*** *( [ ‘index’ , ‘create’ ] );*
    - ***except*** *( [ “destroy”, “store” ] );*
  + ***Actions :-***

GET :  ***index( ) , create( ), show( ), edit( )***

POST :  ***store( )***

PUT/PATCH :  ***update( )***

DELETE : ***destroy( )***

* ***Urls***

***/photos*** *: index( ), store( )*l

**/photos/create** : *create( )*

***/photos/{ photo }/edit*** *: edit( )*

***/photos/{ photo }*** *: show( ) , update( ) , destroy( )*

* ***Naming***
  + *Route :: resource ( ‘photos’ , ‘PhotoController’ ) ->* ***name*** *( [*

*‘create’ => ‘photos.build’*

*] );*

* ***Parameters***
  + ‘*Singularized’* version of *“Photos”* ie *‘photo’*
  + Override *‘photos / { photo }’*  with *‘photos / { admin\_photo }*

*……… ->* ***parameters*** *( [ ‘photos’ => ‘admin\_photo’ ] ) ;*

* ***\_\_contruct ( )***
  + *public function* ***\_\_contruct*** *( UserRepository $user ) {*

*$this->name = $user->name ;*

*}*

* ***Route Parameter & Dependency***
  + *First dependencies then parameters*

*public function store (* ***Request $r*** *,* ***$id*** *) { …… }*

* ***Route caching***
  + *php artisan route:****cache***
  + *php artisan route:****clear***

**REQUESTS**

* *use Illuminate\Http\Request ;*

*public function store (* ***Request $request , $idparameter*** *) { ….. }*

* *$request ->* ***path****( ) => foo/bar*

*->* ***url****( ) =>* [*http://abc.com/foo/bar*](http://abc.com/foo/bar)

*->* ***fullUrl****( ) =>* [*http://abc.com/foo/bar*](http://abc.com/foo/bar)*?id=34&name=umesh*

*->* ***method****( ) => get*

*->* ***isMethod****( ‘post’ ) => ture/false*

* *Input checking for malicious code*
  + *Middlewares ( global )*
    - ***TrimStrings***and ***ConvertEmptyStringsToNull***
* ***Retrieving Input***
  + *$request ->* ***all****( )*

*->* ***input*** *( “name” )*

*->* ***input*** *( “name”, “umesh” )*

*->* ***input*** *( “products.0.name” ) (* If arrays then use *dot operator )*

*->* ***Input*** *( “products.\*.name” ) ( Gives array of all names )*

* ***Retrieving Input ( query string )***
  + *$request ->* ***query*** *( ‘name’ )*

*->* ***query*** *( ‘name’ , ‘Umesh’ )*

*->* ***query*** *( ) ( Gives all query inputs )*

***- Retrieving Input ( dynamic properties )***

*- $request ->* ***name*** *( Any form field names )*

*->* ***age***

* ***Retrieving input ( JSON )***
  + *$request ->* ***input*** *( “user.name” )*
* ***Retrieving Input ( portion only )*** 
  + *$request ->* ***only*** *( [ “name” , “password” ] )*

*->* ***only*** *( “name”, “password” )*

*->* ***except*** *( “name” )*

*-* ***Determine Input ( present / absent )***

*- $request ->* ***has*** *( “name” )*

*->* ***has*** *( [ “name” , “age” ] )*

*->* ***filled*** *( “‘name” ) ( If present and NOT EMPTY )*

*-* ***Old Input***

*-* Handled automatically if built-in solution is used

- Only 1 next request the input is saved

- *$request ->* ***flash*** *( )*

*->* ***flashOnly*** *( [ ‘name’ , ‘age’ ] )*

*->* ***flashExcept*** *( [ ‘password’ ] )*

- Flash & redirect

*- return redirect ( ‘myform’ ) ->* ***withInput*** *( )*

*->* ***withInput*** *( [‘name’ ] )*

- Retrieve Old Input

*- request ->* ***old*** *( ‘name’ )*

*- < iinput tupe=”text” value=”****{{ old ( ‘name’ ) }}****” >*

*-* ***Cookies ( retrieve )***

*-* ***Cookie*** *: : get ( ‘name’ ) & $request ->* ***cookie*** *( ‘name’ )*

*-* ***Files***

* + *Retrieve uploaded files*
    - *$request ->* ***file*** *( “photo” ) ;*

*->* ***photo****;*

*->* ***hasFile*** *( “photo” );*

* + *Valid file*
    - *$request -> file ( “photo” ) ->* ***isValid*** *( );*
  + *File path & extension*
    - *$request -> file ( “photo” ) -> path ( )*

*-> extension ( )*

* + *Store uploaded files*
    - *config /* ***filesystems.php***
    - ***store ( )***
      * *uniqueid as filename*
      * *$request -> file ( “photo” ) ->* ***store*** *( “path” )*

*->* ***store*** *( “path” , ”s3” )*

*->* ***storeAs*** *( “path” , “filename.jpg” , “s3” )*

**RESPONSES**

* ***Create Response*** 
  + *String & Arrays* 
    - *return* ***“hello world”***

***[ 3, 22, 76 ]***

* + *Objects*
    - *return* ***response*** *( “hello” , 200 ) ->* ***header*** *( “content-type” , “text/plain” )*

*->* ***header*** *( “foo” , “bar” )*

* + - *return* ***response*** *( $content ) ->* ***withHeaders*** *( [*

*“H1” => “v1”,*

*“H2” => “v2”,*

*] )*

* *Attach cookies*
  + *Encrypted & signed by* ***EncryptCookies*** *Middleware*
  + *return* ***response*** *( $content ) ->* ***cookie*** *( “name” , “value” , $minutes )*
* ***Redirects***
  + *Global helper*
    - *return* ***redirect*** *( “url” ) ;*

***back*** *( ) ->* ***withInput*** *( ) ;*

* + *Named routes*
    - *return* ***redirect*** *( ) ->* ***route*** *( “RouteName” )*

*->* ***route*** *( “RouteName” , [ ‘id’ => 11 ] )*

* + *Actions* 
    - *return* ***redirect*** *( ) ->* ***actions*** *( “MyController@index” )*

*->* ***actions*** *( “MyController@index” , [ “id” => “value” ] )*

* + *External domain*
    - *return* ***redirect*** *( ) ->* ***away*** *( “*[*https://google.com*](https://google.com)*” )*
  + *With session data*
    - *return* ***redirect*** *( “url” ) ->* ***with*** *( “Key” , “Value” )*
* ***Other response types***
  + *View*
    - *return* ***response*** *( ) ->* ***view*** *( “hello” , $data , 200 ) ->* ***header*** *( “H1”, “V1” )*
    - *Note:* use global **view ( )** helper method if custom header not needed
  + *Json*
    - *return* ***response*** *( ) ->* ***json*** *( [ “name” => “umesh” , “age” => 28 ] )*
    - Content-type is set to *application/json* & array is converted to json
  + *Download*
    - *return* ***response*** *( ) ->* ***download*** *( “path” , “name.jpg” , $headerOptional )*

*->* ***download*** *( “path” )*

*->* ***download*** *( “path” ) ->* ***deleteFileAfterSend*** *( )*

* + *Open in browser ( pdf / images )*
    - *return* ***response*** *( ) ->* ***file*** *( “path” )*

**VIEWS**

* *resources / views / welcome****.blade.php***
* *return* ***view*** *( “welcome” , [ “name” => “james” ] )*

***view*** *( “welcome” ) ->* ***with*** *( [ “name” => “james” ] )*

* *Variable shared with all views* 
  + *boot ( ) { // AppServiceProvider* ***or***  *MyProvider*

***View*** *: :* ***share*** *( ‘key’ , ‘value’ )*

*}*

* ***Composer***
  + *Composer* is a *class* *method* called during rendering of *view*
  + Location :  *“app/Http/View/Composers/ProfileComposer”*
  + *Register composer*

*boot ( ) { // ViewServiceProvider*

*// File based & ‘profile’ is a view file ‘profile.blade.php’*

***View*** *: :* ***composer*** *( ‘profile’ , ‘App\Http\View\Composers\ProfileComposer’ )*

*// Closure based*

***View*** *: :* ***composer*** *( ‘profile’ , function ( ) { ……….. } )*

*// For Muliple views : [ ‘profile’ , ‘home’ , ‘welcome’ ]*

*// For all views : “\*”*

*}*

* *ProfileComposer*
  + ***compose*** *( ) called whenever profile view is rendered & ‘$count’ variable available*
  + ***compose*** *( View $view , UserRepo $users ) {*

*$view -> with ( ‘count’ , $users -> count ( ) )*

*}*

**URL GENERATIONS**

* *Basics*
  + ***url*** *( “posts/11” ) // http/s://example.com/posts/11*
  + ***url*** *( ) ->* ***current*** *( ) // without query*

*->* ***full*** *( )*

*->* ***previous*** *( )*

* *Named Route*
  + *$url =* ***route*** *( “RouteName” , [ ‘id’ => 11 ]*
* *Signed urls*
  + ***URL*** *: :* ***signedRoute*** *( “unsubscribe” , [ “user” => 11 ] )*

: : **temporarySignedRoute** ( ‘unsubscribe”, now( ) -> addMinutes (30), [ “user” => 11 ] )

* *Validation of sign ( can be a middleware )*
  + *$request ->* ***hasValidSignature*** *( )*
* *Controllers actions*
* *$url =* ***action*** *( “homeController@show” , [ ‘id’ => 11 ] )*

**SESSIONS**

* ***Introduction***
  + HTTP is stateless protocol
  + *config/session.php* 
    - Options : *file, cookie, database, redis, memcached*
  + *database* (table)
    - *php artisan session:table*
  + *redis*
    - *composer require predis/predis*
    - See *config/database.php*
* ***Using Sessions***
  + ***session ( )*** *OR* ***$request -> session ( )***
  + *Retrieve key value*
    - *$request ->* ***session*** *( ) ->* ***get*** *( “key” )*

*->* ***get*** *( “key” , “default” )*

*->* ***all*** *( )*

* + - ***session*** *( “key” )*

*( “key” , “default” )*

* + *Store new key*
    - *$request -> session ( ) ->* ***put*** *( ‘key’ , ‘value’ )*
    - ***session*** *( [ “key” => “value” ] )*
  + *Push / Pull* 
    - *$request -> session ( ) ->* ***push*** *( ‘key’ , ‘value’ )*

*->* ***pull*** *( ‘key’ , ‘default’ ) // returns key or default value*

* + *has / exits*
    - *$request -> session ( ) ->* ***exists*** *( “key” ) // value can be null*

*->* ***has*** *( “key” ) // value Not null*

* + *Flash data*
    - *$request -> session ( ) ->* ***flash*** *( “status” , “Successful” )*

*->* ***reflash*** *( )*

*->* ***keep*** *( [ ‘name’ , ‘age’ ] )*

* + *Remove* 
    - *$request -> session ( ) ->* ***forget*** *( “key” )*

*-> forget ( [ “key” , “key2”....... ] )*

*->* ***flush*** *( )*

* + *Regenerate session id ( Done automatically on user login )*
    - *$request -> session ( ) ->* ***regenerate*** *( )*

**VALIDATION**

* ***Validate ( )***
  + If validation fails response is automatic
    - *http* for *http* *request* (redirect to previous)
    - *json* for *ajax request*
  + *$valid = $request ->* ***validate*** *( [*

*“title” => “required | max:255” ,*

*“age” => “required”*

*] );*

* ***‘bail’*** *rule* to stop checking for next rule after a rule fails
* *Null values*
  + Uses middlewares : *TrimString* & *ConvertEmptyStringsToNull*
  + *null values* gives validation error. So use *‘nullable’* rule
    - *‘date’ => ‘****nullable*** *| date’*
* ***Form Request Validation***
* *Create*
  + *php artisan* ***make:request*** *StoreBlogPost*
* *Rules* 
  + ***rules*** *( ) {*

*return [*

*‘name’ => ‘required | max:250’ ,*

*‘age’ => ‘required’*

*] ;*

*}*

* *Authorize* 
  + *Route : : post ( ‘comment/{ commentPara } , …..)*
  + ***authorize*** *( ) {*

*// return true if user is authorized to comment*

*$comment = Comment : : find ( $this -> route ( ‘commentPara’ ) );*

*return $comment && $this -> user( ) -> can( ‘update’ , $comment ) ;*

*// or*

*// return true for always authorized*

*return true*

*}*

* *Custom error messages*
  + ***message*** *( ) {*

*return [ ‘name.required’ => ‘Invalid or missing name. Please check again’ ];*

*}*

* *Use in controller*
  + *public function store* ( *StoreBlogRequest $request , $id ) {*

*$valid = $request -> validate ( );*

*}*

* ***Manually create validators*** 
  + Not much usefull. see docs.
* ***Errors Messages***
  + *$****errors*** *->* ***first*** *( ‘email’ ) // 1st error for email field*

*->* ***get*** *( ‘email’ ) // all errors for email field*

*( ‘projects.\*’ ) // all errors for array*

-> ***all*** *( ) // all errors all fields*

*->* ***has*** *( ‘email’ ) // check if error for email field*

*->* ***any*** *( ) // check if any error in any field*

**ERROR HANDLING**

* *APP\_DEBUG* option in *.env*
  + true = development
  + false = production
* ***Exception Handler Class***
  + *App\Exception\Handler*
  + ***report ( )***
    - *log or send* exceptions to *bugsnag/sentry*
    - Passes exception to *base class* for logging
    - Custom way to handle any exception
      * *if ( $exception instanceof CustomException ) { …….. }*
    - *report ( )* ( helper method )
      * Reports exception & continues execution
      * *try {*

*# code…..*

*} catch ( Exception $e ){*

*report ( $e );*

*report false ;*

*}*

* + *Ignore Exceptions*
    - *protected* ***$dontreport*** *= [*

*\Illuminate\Auth\AuthenticationException : : class ,*

*\Illuminate\Database\Eloquent\ModelNotFoundException : : class*

*]*

* + ***render ()***
    - Generates *http response*
    - *if ( $exception instanceof CustomException ) {*

*return response ( ) -> view ( ‘errors.custom’ , [ ] , 500 )*

*}*

* ***Custom class*** 
  + *class MyException extends Exception {*

*report ( ) { ……. } ;*

*render ( $request ) { …….. }*

*}*

* + ***HTTP Exceptions***
    - ***abort*** *( 404 )*

*( 403, “Unauthorized action” )*

* + - Views to edit
      * *views/errors/404.blade.php*
      * *php artisan vendor:publlish --tag=laravel*

**LOGGING**

* Uses *Monolog* library
* *config/logging.php*
  + Channel drivers
    - *stack (multi-channel) - syslog*
    - *single - errorlog*
    - *daily - monolog*
    - *slack - custom*
* see docs for details

**BLADE**

* ***Introduction***
  + *Layout Master* *(layouts/app.blade.php)*
    - ***@yield*** *( ‘title’ )*
  + ***@section*** *( “sidebar” )*

*……..*

***@show*** *// both define & yield section*

* *Extending layout* *(child.blade.php)*
  + ***@extends (*** *‘layouts.app’)*
  + ***@section*** *( ‘title’ , “This is title” )*
  + ***@section*** *( ‘title’ ) …….………. @****endsection***
  + ***@section*** *( “sidebar” )*

***@parent***

*<p> Appended after parent’s content </p>*

***@endsection***

* *Usage : return* ***view*** *( “child’ )*
* ***Components & slots***
  + *Component**( alert.blade.php )*
    - *<div class=”alert” >*

*<h1> {{ $msg }} </h1>*

*{{* ***$slot*** *}}*

*</div>*

* + *Usage*
    - ***@component*** *( ‘alert’ )*

***@slot*** *( "msg” )*

*……..*

***@endslot***

***…..****This comes in $slot variable*

***@endcomponent***

* *@component ( “alert” , [ ‘foo’ => ‘bar’ ] ) // Pass data directly*
  + *Aliasing*
    - *views/components/alert.blade.php*
    - *boot ( ) { // AppServiceProvider*

*Blade : :* ***component*** *( “componets.alert” , “alert” ) ;*

*}*

* + - *Usage*
      * ***@alert*** *……….* ***@endalert***
* ***Display data***
  + ***{{*** *$name* ***}} //*** *variable*

***{{*** *time ( )* ***}}*** *// Function / method*

***{!!*** *$name* ***!!}*** *// Unescaped data*

* *Unrendered* ***(*** *Vue / React js variables )*
  + ***@{{*** *name* ***}}***
  + ***@verbatim***

*<p> {{ name }} </p>*

***@endverbatim***

* + *JSON*
    - *<script> var a =* ***@json*** *( $array) ; </script>*
* ***Control structures***
* *IF*

*@****if*** *(condition)*

*…….*

*@****elseif*** *(condition)*

*…….*

*@****else***

*…...*

*@****endif***

* *Auth*
  + ***@auth*** *…….…*  ***@endauth //*** *also @auth**( “****guardname****” )*
  + ***@guest*** *……...…* ***@endguest*** *// also @guest ( “****admin****” )*
* *Switch*
  + ***@switch*** *( $i )*

*@case(1)*

*………*

*@break*

*@case(2)*

*………*

*@default*

*………*

***@endswitch***

* *Loop*
  + - ***@for*** *($a = 1 ; $a < 6 ; $a++ )*
    - **@foreach** *( $users as $user )*
    - ***@while*** *( condition )*
    - ***@condition*** *&* ***@break***
    - *$loop*
      * ***$loop*** *->* ***first***

*->* ***last***

*->* ***index***

*->* ***parent -> first***

* *Comment* 
  + **{{--** This is a comment. Not shown in rendered html code **--}}**
* *Php code*
  + ***@php*** *……….. @endphp*
* ***Forms***
* ***@csrf***
* ***@method*** *( “put” )i*
* ***Include partial views***
  + ***@include*** *( “partials.profile” , [ “foo” => “bar” ] )*
  + *Aliasing*
    - *boot ( ) { // AppServiceProvider*

*Blade : : include ( “partials.profile” , “profile” ) ;*

*}*

* *Usage : @profile( [ “foo” => “bar” ] )*
* ***Stacks***
* ***@stack*** *( “script” )*
* *Usage :*
  + ***@push*** *( “script” ) …….. @endpush*
* ***Extend Blade***
* *boot ( ) { // AppServiceProvider*

*Blade : :* ***directive*** *( “datetime” , function ( $t ) {*

*return “ <?php echo $t -> format ( “m/d/y H:i “ );*

*} );*

*}*

* *Usage :* ***@datetime*** *( $time )*

***LOCALIZATION***

* *see docs*

**FRONTEND SCAFFHOLDING**

* *Package.json ( Bootstrap, Axios, Vue, Jquery, etc )*
* *npm* ***install***
* *npm run* ***dev***

***watch***

* Laravel Mix configs in *Webpack.mix.js*
* *Remove* *( Bootstrap, Vue )* 
  + *php artisan preset none*
* ***css***
  + ***resources****/sass/ TO* ***public****/css*
* ***js***
  + ***resources****/js TO* ***public/****js*
  + *Vue*
    - *resources/js/components/ExampleComponent.vue*
    - Single file has both *html & js*
    - ***Vue.component (*** *‘example****’ , require ( ‘****./components/ExampleComponent.vue****’ ) )***
    - *Usage :- < example > …… </ example >*
  + *React*
    - *php artisan preset react*

**COMPILING ASSETS (MIX)**

* *Mix* make *Webpack* easier using *Webpack.mix.js*
* ***mix****.****js*** *( “resources/js/app.js” , “public/js” )*

*.****sass*** *( “resources/css/app.scss” , “public/css” )*

* Needs *nodejs* with *npm* installed
* Install packages :- *npm* ***install***
* Run
  + *npm run* ***dev | production | watch | watch-poll***
* ***Working with js*** 
  + *Vendor extraction*
    - *mix.js ( …. )****.extract*** *( [ “vue” ] )*
    - Gives 3 files ***manifest.js , vendor.js , app.js (****sequence must be same in <html>)*
    - *why ? If updates are frequent then avoid compiling vendors again & again.*
* ***Version / Cache Busting***
  + Appends a *hash* to filename
  + *mix.js ( …….. )* ***. version( )***
  + *usage : <script src = ‘ {{* ***mix*** *( “js/app.js” ) }} ‘ >*
  + Only in production. No use in development.
    - *if ( mix.inProduction ( ) ) {*

*mix.version ( ) ;*

*}*

* **BrowserSync**
  + Moniters files for any changes & if changed then browser is refreshed automatically.
  + *mix.****browserSync*** *( “domain.test” )*
  + Run :- *npm run* ***watch/watch-poll***

***--------SECURITY--------***

**AUTHENTICATION**

* ***Introduction***
  + *php artisan make:auth*
  + *config/auth.php*
  + Based on *guards ( web/api ) & providers ( eloquent/query with user table )*
  + *App\Users* table model is used
* ***Auth Quickstart***
  + *Controllers*
    - *HomeController*
    - *App\Http\Controllers\Auth* 
      * *Register, Login, ForgotPassword, ResetPassword*
      * ***$redirectTo***& ***redirect( )*** (Also RedirectIfAuthenticated middleware )
  + *Routes : Auth : : routes ( )*

*( [ “register” => “false” ] ) // If not needed*

* + *Views : resources/views/auth & layouts*
  + *Username*
    - “*email*” field is default
    - If others then use this method.in *loginController*
      * *public function* ***username*** *( ) { return “username” ; }*
  + *Guard*
    - *In Login , Register, ResetPassword controllers*
    - *public function* ***guard*** *( ) { return Auth : : guard ( “name” ) ; }*
  + *Create & Validation Of Users*
    - *RegisterController* has methods ***validate( )*** *&* ***create( )***
  + *Retrieve auth users*
    - *Auth : :* ***user*** *( )*

*: :* ***id*** *( )*

*: :* ***check*** *( )*

* *Login Throttling*
  + - Based on unique *username & IP address*
* ***Manually auth users***
  + Incase manual we must ignore default controllers
  + ***attempt ( )***
    - *$credentials = $request -> only ( ‘email’ , ‘password’ ) ;*

*if ( Auth : :* ***attempt*** *( $credentials ) ) {*

*return redirect ( ) -> intended ( “dashboard” ) ;*

*}*

* Specific guard *: Auth : :* ***guard*** *( ‘admin’ ) -> attempt ( $credentials )*
  + *Logout : Auth : :* ***logout*** *( )*
  + *Remember me* 
    - *remember\_token* in *users* table as hash values ( *LoginController* already has this *)*
    - *Auth : : attempt ( $credentials , $****remember*** *)*
  + *Other auth ways*
    - *User Instance*
      * Needs *Authenticable* contract *( App\User* implements this contract )
      * *Auth : : login ( $****user*** *)*

*( $user ,* ***true*** *) //* remember me checked

* ***Http Basic auth***
  + *Prompt* is used. So dedicated *login*  view pages required
  + Use ***auth.basic*** middleware
* ***Add Custom Guards & User Providers*** 
  + *see docs*
* ***Events***
  + *EventServiceProvider*
  + *Registered , Login , Attempting , Authenticated ,*

*Failed , Logout , Lockout , PasswordReset*

**API AUTH PASSPORT**

* see docs.

**AUTHORIZATION**

* Authorize a user to use a given resource.
* *Gates(closure) & Policies(logic group)* are like *Routes & Controllers*
* *Both Gates &Policies* return *false* if request is not from logged-in user. Also see Guest below.
* ***Gates***
  + *Writing*
    - In *App\Providers\AuthServiceProvider*
    - Always ***$user*** instance as first argument. Rest are optional arguments.
    - *boot ( ) {*

*$this -> registerPolicies ( );*

*# closure based*

*Gate : :* ***define*** *( “update-post” ,* ***function*** *( $user , $post )*

*return $user -> id == $post -> user\_id ;*

*) ) ;*

*# class@method*

*Gate : :* ***define*** *( “update-post” , “****App\Policies\PostPolicy@update”*** *) ;*

*}*

* + - *Resource Gates*
      * *Gate : :* ***resource*** *( “posts” , “App\Policies\PostPolicy” ) ;*
      * *Total 4 gates => posts****.view*** *[ view, create, update, delete ]*

*PostPolicy****@view*** *[ view, create, update, delete ]*

* + *Authorizing actions*
    - *Gate : :* ***allows*** *( “update-post” , $post ) // true/false*

*: :* ***denies*** *( “update-post” , $post ) //* note : *$user* is not needed to pass

* + - *Gate : :* ***forUser*** *( $user ) -> allows ( “update-post” , $post )*
* ***Policies***
  + Uses *model or resource*
  + *Stored in app/Policies*
  + *php artisan make:policy PostPolicy --model=Post*
  + *Register*
    - In *AuthServiceProvider* register model-policy relations
    - *protected* ***$policies*** *= [*

*Post : : class => PostPolicy : : class ,*

*]*

* + *Writing*
    - In *Policies*/*PostPolicy* define methods with any names.
    - *update ( User $user , Post $post ) {*

*return $user -> id == $post -> user\_id ;*

*}*

* + - Sometimes other model instance is not needed

*create ( User $user ) { ……. }*

* + - *Guest*
      * Login is must to use gates & policies
      * Optional $user can work for guest
      * *update (* ***?****User $user , Post $post ) { ….. }*
    - *Admin*
      * Must returns true for all policy methods. Use ***before( )***
      * *public function* ***before*** *( $user , $ability ) {*

*if ( $user -> isSuperAdmin ( ) ) {*

*return true ;*

*}*

*}*

* ***Authorizing actions using policies***
  + *Via User Model*
    - *$user ->* ***can*** *( “update” , $post ) // true / false*

***cant*** *( “update” , $post )*

* + - First call *policy@method*. If Not found then call *Gate closure*
    - *Create*
      * *$user -> can ( “create” ,* ***Post : : class*** *) // $post* not needed
  + *Via Middleware*
    - *‘can’ => ‘Illuminate\Auth]Middleware\Authorize’ // key in Http\Kernel*
    - *see docs*
  + *Via Controller Helpers*
    - ***authorize ( )***method in base *Controller*
    - *update ( Request $request , Post $post ) {*

*$this ->* ***authorize*** *( “update” , $post ) ;*

*# Run if true*

*}*

* *create ( Request $request ) {*

*$this ->* ***authorize*** *( ‘create’ ,* ***Post : : class*** *) ;*

*// run if true*

*}*

* + *Via Blade Templates*
    - ***@can*** *&* ***@cannot***
    - *@can ( “update” , $post )*

*……..*

*@elsecan ( “create” , App\Post : : class )*

*…….*

*@endcan*

**EMAIL VERIFICATION**

* ***Introduction***
  + *Model preparation*
    - *App\User* must implement *Illuminate\Contracts\Auth\MustVerifyEmail*
* ***Database Considerations***
  + *users* table must have *email\_verified\_at*
* ***Route***
  + *Auth\VerificationController* has logic for sending emails
  + *Auth : : routes ( [* ***‘verify’ => true*** *] )*
  + Use ***verify*** middleware which is *Illuminate\Auth\Middleware\EnsureEmailIsVerified*
* ***Views***
  + *resources/views/auth/verify.blade.php*
* ***After verifying emails***
  + *protected $****redirectTo*** *= “/dashboard” // in VerificationController*
* ***Events***
  + *Verified* event

**ENCRYPTION**

* *OpenSSL* provides *AES-256 / AES-128*
* All encrypted values are signed with MAC (Message Authentication code)
* *php artisan key:generate // Gives AES-256-CBC key in .env*
* ***Using the encrypter***
  + ***encrypt*** *( $request -> secretmsg )*
  + Uses ***serialize***during encryption for *objects & arrays*
  + ***Crypt*** *: :* ***encryptString*** *( ‘hello’ ) ;*
    - *: :* ***decryptString*** *( $encryptedKey ) ;*

**HASHING**

* *Bcrypt & Argon2* algorithms are available for passwords. *Bcrypt* is default for laravel
* *config/hashing.php*
* ***Hash*** *: :* ***make*** *( $request -> password )*

*: :* ***check*** *( “plainText” , $hashedPassword )*

*: :* ***needsRehash*** *( $hasedPassword ) // true / false*

**RESET PASSWORD**

* *User* must use trait *Illuminate\Notifications\Notifiable*
* *Reset tokens* expires after 1 hour. See *expire* option in *config/auth.*php
* ***Database*** 
  + *App\User* must implements *Illuminate\Contracts\Auth\CanResetPassword*
  + DB table is already added in migration.
* ***Route***
  + *Auth\ForgetPasswordController & Auth\ResetPasswordController*
* ***Views***
  + *resources/views/auth/passwords*
* ***After Reset*** 
  + Browser visit  *“/password/reset”*
  + *$redirectTo* property after reset of password.

**ARTISAN CONSOLE**

* ***Introduction***
  + *php artisan* ***list***

***help <cmd>***

***tinker***

***make:command MyCmd***

* ***Writing commands***
  + Custom commands are stored in  *app/Console/Commands*
  + *php artisan make:command SendEmails*
  + *SendEmails* *class based*
    - ***$signature*** *= “email:send { user } “*
    - ***$description*** *= “Send drip emails to a user”*
    - ***handle*** *( DripEmailer $drip ) {*

*$drip -> send ( User : : find ( $this -> argument ( “user” ) ) ) ;*

*}*

* + *Closure based*
    - *routes/console.php* is where closure based commands are defined
    - Not *http* routes but *console* based entry points
    - *Artisan : :* ***command*** *( “build { project } ,* ***function*** *( $project ) {*

*$this -> info ( “Building { project } “ ) ;*

*) ) ->* ***describe*** *( “Build the proect “ ) ;*

* + - *Artisan : :* ***command*** *( “email : send { user } “,* ***function*** *( DripEmailer $drip , $user ) {*

*$drip -> send ( User : : find ( $user) ) ;*

*}*

* ***Defining Input expectations***
  + *Arguments*
    - *$signature = “email : send {* ***user*** *} “*

*{ user****?*** *) // Optional*

*{ user****=foo*** *} // Defualt*

* + *Options* 
    - 2 types based on option *value* received or not .
    - If not option *value* is not received then it acts as *true/false*
    - *$signature = “ email : send { user } {* ***--queue*** *} “ // true / false*

*{ --queue****=*** *} // value is needed*

*{ --queue=default } // default value*

*{ --****Q****|queue } // shortcut*

* ***Command IO***
  + *Retrieving input* 
    - *In handle ( ) method*
      * *$userId = $this ->* ***argument ( ) //*** *All args as array*

*(* ***“user”*** *) // only user*

* + - * *$queueName = $this -> option* ***( ) //*** *All options as array*

*(* ***“queue”*** *) // only queue*

* + - *Prompt for input*
      * *$name = $this ->* ***ask*** *( “what is your name” )*

***secret*** *( “type password” )*

* + - * *$this ->* ***confirm*** *( “Do you wish to continue ?“ ) // true if “y” / “yes”*
  + *Display output*
    - *In handle ( ) method*
      * *$this ->* ***info*** *( “Display this on the screen” )*
    - ***info*** *(green) ,* ***error*** *(red) ,* ***line*** *(plain uncolored) ,* ***question*** *,* ***comment***
  + *Table layout / Progress bars* 
    - *See docs*
* ***Registering command***
  + *app/Console/Commands* directory’s all commands are automatically registered by kernel
* ***Programmatically Executing Commands***
  + To execute cmd from say *controller or route* and not in *CLI (command line interface)*
  + *$exitCode =* ***Artisan*** *: :* ***call*** *( “email:send” , [ “user” => 12 , “--queue” => “default” ] )*

***queue*** *( …. ) -> onConnection ( ‘redis’ ) -> onQueue ( “commands” )*

**BROADCAST ( Pusher )**

* **Realtime / live updating user interfaces**

**CACHE**

* ***Config***
  + *config/cache.php =>* drivers [ *File, Database, Redis, Memcache ]*
  + See Docs for driver’s config.
* ***Cache usage***
  + *Retrieve* 
    - *Cache : :* ***get*** *( ‘key’ , ‘default’ )*

*: : ( ‘key’ ,* ***function*** *( ) { …….. } )*

*: :* ***store*** *( ‘redis’ ) -> get ( ‘key’ )*

*: :* ***has*** *( ‘key’ ) // check*

*: :* ***increment*** *( ‘key’ ) // ++ / -- integer value*

*: : ( ‘key’ , $****amount*** *)*

* + *Retrieve & store*
    - Retrieve if present or add key in cache via function return
    - *Cache : :* ***remember*** *( ‘key’ , $minutes , function ( ) { …... } )*

*: :* ***rememberForever*** *( ‘key’ , function ( ) { ….. } )*

* + *Retrieve & delete*
    - *Cache : :* ***pull*** *( ‘key’ )*
  + *Storing* 
    - *Cache : :* ***put*** *( ‘key’ , ‘value’ , $minutes )*

*: :* ***add*** *( ‘key’ , ‘value’ , $minute )*

*: :* ***forever*** *( ‘key’ , ‘value’ )*

* + *Remove*
    - *Cache : :* ***forget*** *( ‘key’ )*

*: :* ***flush*** *( )*

* + *Atomic Locks ( redis or memcache only )*
    - Used to avoid race conditions
    - *if ( Cache : :* ***lock*** *( ‘foo’ , 10 ) -> get ( ) ) {*

*// lock acquired for 10 seconds*

*Cache : : lock ( ‘foo’ ) ->* ***release*** *( ) ;*

*}*

* + *Cache helper* 
    - ***cache*** *( ‘key’ )*

*( [ ‘key’* ***=>*** *‘value’ ] ,$minutes )*

* ***Cache tags***
  + *Note :* Not supported for *file & database* drivers
  + *Tag* related items which helps in using & removing them together
  + *Store*
    - *Cache : :* ***tags*** *( [ ‘tag1’ , ‘tag2’ ] ) ->* ***put*** *( ‘key’ , $key , $minutes )*
  + *Access*
    - *Cache : :* ***tags*** *( [ ‘tag1’ , ‘tag2’ ] ) ->* ***get*** *( ‘key’ )*
  + *Remove*
    - *Cache : :* ***tags*** *( [ ‘tag1’ , ‘tag2’ ] ) ->* ***flush****( )*
* ***Adding custom cache drivers ( MongoDB )***
  + *see docs*
* ***Events***
  + *CacheHit , CacheMissed , KeyForgotten , KeyWritten*

**COLLECTION**

* Wrapper for arrays. *Eloquent* results are always collections
* Are *Immutable* meaning every method ( *map, reject etc* ) in chain returns new collection instance.
* To Uppercase and remove null values from array
  + *$collect =* ***collect*** *( [ ‘a’ , ‘b’ , null , ‘c’ ] ) ->* ***map*** *( function ( $n ) { return strtoupper ( $n ) ; } )*

*->* ***reject*** *( function ( $n ) { return empty ( $n ) ; } ) ;*

* ***Extending***
  + ***Collection*** *: :* ***macro*** *( “toUpper” , function ( ) { // In service provider*

*return $this -> map ( function ( $n ) {*

*return Str : : upper ( $value ) ;*

*}*

*} ) ;*

* Usage : *$collect -> toUpper ( ) ;*
* ***Available Methods***
  + all ( ) , map ( ) , take ( ) , count ( ) , dd ( ) , min ( ) , toArray ( ) , get ( ) , isEmpty ( ) , etc…
  + *See Docs*. Lots of methods.

**EVENTS**

* *app/Events* ***&*** *app/Listeners*
* 1 Event can have many Listeners
* ***Register***
  + *protected $listen = [ // EventServiceProvider*

*‘App\Events\OrderShipped’ => ‘App\Listeners\SendEmail’ ;*

*]*

* + All registered events & listeners files will be automatically created by cmd
    - *php artisan event:generate*
  + *Manual way ( closure based )*
    - *boot ( ) { // EventServiceProvider*

***Event : : listen*** *( ‘event.name’ , function ( $foo , $bar ) { ...... } );*

*}*

* ***Define Events***
  + ***\_\_construct*** *( Order $order ) { // Events/OrderShipped*

*$this -> order = $order ;*

*}*

* + Has no logic but only data of order instance purchased.
* ***Define Listeners***
  + ***handle*** *( OrderShipped $event ) { ……. } // Listeners/SendEmail*
  + To stop propogation to other listeners of that events ***return false;***
* ***Queued Event Listners***
  + Queuing is useful if slow tasks like *sending email, http request, etc*
  + Configure & start queue process on server to use queues for listeners
  + Usage: *class SendEmail implements* ***ShouldQueue*** *{ ........ }*
  + Extra config : *public* ***$connection*** *= ‘sqs’ ;*

***$queue*** *= ‘listeners’ ;*

***$delay*** *= ‘60 ;*

* + Fails: *public function* ***failed*** *( OrderShipped $event , $exception ) { …. }*
* ***Dispatch Events***
  + ***event*** *( new OrderShipped ( $order ) )* ; // *$order* instance
* ***Event Subscribers***
  + A class which defines many event handlers.
  + *namespace App\Listeners ;*

*class UserEventSubscriber {*

*public function* ***onUserLogin*** *( $event ) { }*

*public function* ***onUserLogout*** *( $event ) { }*

*public function* ***subscribe*** *( $events ) {*

*$events ->* ***listen*** *( ‘Illuminate\Auth\Events\Login’ ,*

*‘App\Listeners\UserEventSubscriber@****onUserLogin****’ ) ;*

*$events ->* ***listen*** *( ‘Illuminate\Auth\Events\Logout’ ,*

*‘App\Listeners\UserEventSubscriber@****onUserLogout****’ ) ;*

}

}

* + Register ( In *App\Providers\EventServiceProvider* )
    - *protected* ***$subscribe*** *= [ ‘App\Listeners\UserEventSubscriber’ ]*

**FILE STORAGE**

* fLYSYSTEM package
* Local / Amazon S3 / Rackspace
* ***storage/app/public***
* Create symboliclink from *public/storage* to *storage/app/public*
* *php artisan storage:link* ----> Create symbolic link
* ***asset****(‘storage/file.txt)*

**MAIL**

* ***Mailgun.com***  ------> 10000 mails/ month FREE
* Swiftmailer php library
* SMTP / Mailgun / SparkPost / Amazon SES / php’s mail() & sendmail()
* Local or cloud based service
* Mailgun, SparkPost --- faster than ---- SMTP
* *Composer require guzzlehttp/guzzle*
* *config/mail.php ----->* select driver
* config/services.php ------> secret-key && domain
* *app/Mail* ------each email sent is aMailable class
* *php artisan make:mail MyMail*
* *Public function build(){*

*return $this->****from****(“*[*me@ex.com*](mailto:me@ex.com)*”)->****view****(“myview”)*

*}*

* Global ***from()***  method in *config/mail.php*
* ***attach****(‘path/to/file’)*
* $*user = User::findOrFail( $userId );*

***Mail::to*** *( “*[*abc@exam.com*](mailto:abc@exam.com)*” )->****send****( new MyMail( $user ) );*

* ***Mail::to*** *( “*[*abc@exam.com*](mailto:abc@exam.com)*” )*

*->****cc****($ccUsers)*

*->****bcc****($bccUsers)*

*->****send****( new MyMail( $user ) );*

* Increased response time ----> use queues

**NOTIFICATIONS**

**PACKAGE DEVELOPMENT**

**DATABASE**

* Raw / query builder / eloquent ORM
* Transactions (**All or none**)

*DB::****transaction****(function () {*

*DB::table(‘users’)->update([ ‘votes’ => 1]) ;*

*DB::table(‘post’)->delete();*

*} , 5 ); -----> 5 = attempts after deadlock*

**QUERY BUILDER**

* *DB::table(‘users’)->get()*
* *DB::table(‘users’)->where(‘name’ , ‘john’ )->****first****() ;*
* *DB::table(‘users’)->where(‘name’ , ‘john’ )->****value****(‘name’)*
* *->****pluck****(‘anyCol’);*

**PAGINATION**

* *paginate( ) ---->*on query builder / eloquent
* *DB::table(‘users’)->****paginate****(15) ;*
* *DB::table(‘users’)->simpleP****aginate****(15) ;*
* *App\User ::* ***paginate****(15) ;*
* *App\User ::* ***simplepaginate****(15) ;*
* *{{ $user->****links****() }}*

**MIGRATION**

* *php artisan make:migration create\_users\_table --create=users*
* *database/migrations*
* ***php artisan migrate***
* *php artisan migrate:refresh*
* *php artisan migrate:refresh --seed*

**SEEDS**

* *php artisan make:seeder UsersTableSeeder*
* *DatabaseSeeder.php*
* *php artisan db:seed*

**REDIS**

* Advanced key value store
* Keys can be ----> strings, hashes, lists, sets, sortedsets
* Composer require predis/predis
* Use redis in *config/database.php*

**ELOQUENT**

* ORM
* MODELS
  + *php artisan make:model User --migration*
  + *php artisan make:model User --m*
  + *protected*  ***$table*** *= ‘users” ;*
  + *Protected* ***$dateFormat*** *= ‘U’*
* *User::* ***find****(1) ;*
* *User::* ***find****( [ 4, 6, 7 ] ) ;*
* *User::where( ‘active’ , 1 )->****first****( );*
* *User::* ***findOrFail****(1) ; -----> 1st row with id = 1*
* *User:: where( ‘legs’ , ‘ ’>’ , 100 )->****firstOrFail****(1) ;*
* *Inserts*
  + *$user = new User ();*

*$user->name = $request->name ;*

*$user->****save****( );*

* + *$user = User::find (11);*

*$user->name = “hello”;*

*$user->save( );*

* *Protected* ***$fillable*** *= [‘name’];*
* *$user = User::****create****( [ ‘name’ => ’ myname’ ] );*
* *$user->****fill****( [ ‘name’ => ‘myname’ ]);*
* *User::****firstOrCreate****( [ ‘name’ => ’ myname’ ] );*
* *User::****firstOrNew****( [ ‘name’ => ’ myname’ ] );*
* *User::****updateOrCreate****( [ ‘name’ => ’ myname’ ] , [‘age’ => ‘26’ ]) ;*
* *User::find( 11 )->****delete****();;*
* *User::****destroy(11)****;;*
* *User::****destroy( [*** *11 , 45 , 33 ]* ***)****;;*

**RELATIONSHIP**

* Methods in model

1 - 1

* *$this->****hasOne(*** *“Car’* ***); ------> inside User.php model file***
* *$this->****belongsTo(*** *“User’****); ------> inside Car.php model file***

1 - \*

* *$this->****hasMany(*** *“Car’* ***); ------> inside User.php model file***
* *$this->****belongsTo(*** *“User’****); ------> inside Car.php model file***
* *$this->****belongsToMany(*** *“Car’* ***); ------> inside User.php model file***
* *$this->****belongsToMany(*** *“User’****); ------> inside Car.php model file***

**ACCESSOR / MUTATORS**

* Like Get / set
* Methods inside model
* *Public function* ***get****FirstName****Attribute*** *($value) { ---->first\_name* is column name

*Return ucfirst($value) ;*

*}*

* *Public function* ***set****FirstName****Attribute*** *() {*

*$this->attributes[ ‘first\_name’ ] = strtolower ($value);*

*}*