**INSTALLATION**

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

- ROOT to public folder

- check config folder

> timezone

> mysql utf8\_unicode\_ci

> locale lang

- give folder/file permissions[ use sh-start.sh file ]

- .env key

- apache enable mod\_rewrite module

**CONFIGURATION**

* .env conifguration
* *config( ‘app.timezone’ );*
* *Php artisan config:cache*
* Maintenance mode
  + *php artisan down* -- no queued jobs handled
  + *Php artisan up -- queue resume*
* Envoyer - Alternative to maintenance mode

**REQUEST LIFECYCLE**

* Entry point public/*index.php*
  + Loads composer *autoload*
  + New *bootstrap/app.php* instance of Laravel Application
  + Laravel creates new instance *application/service-container*
  + Incoming request
    - HTTP/console kernel
    - HTTP KERNEL
      * Extends *Illuminate\Foundation\Http\Kernel*
      * Run bootstrappers/tasks before request handling
        + *register()*  service providers from *config/app.php*
        + *boot()*
      * Dispatched to *router*
      * Middlewares
      * *handle()* ----- receives request & gives response

**SERVICE CONTAINER**

* Bindings registered within service container
* No need to *bind* classes if do not depend on interfaces
* Inside provider use ***$this->app*** property to access container
* Register bindings

***bind()***

* + Passing class/interface name
  + Function (a closure) gets *$app*  which is
  + Function returns **new** class instance **each time**
  + Eg: *$this->app->bind( “path\MyClass” , function($app){*

*return new path\MyClass ($app->make(“MyOtherClass”));*

*});*

***singleton()***

* Resolved only once
* Returns **same** instance **each time**
* Eg: $this->app->singleton(“path\MyClass”, function( $app){

return new path\MyClass($app->make(“MyOtherClass”));

});

***instance()***

* Bind existing instance to container
* Eg : $myclass = new MYClass(new OtherClass) ;

$this->app->instance(“MyClass”, $myclass);

**Binding primitives**

* If class requires other class instance and primitive(int, string etc) value too
* Eg: $this->app->when(“path\MyClass”)

->needs(“$anyVariable”)

->give($value);

**SERVICE PROVIDER**

* Bootstraped = registering things
* ***config/app.php*** *---- $****providers*** *array*
* Some are deferred services ie loaded when required
* Extends *Illuminate\Support\ServiceProvider*
* ***register()***  *only* binding(nothing other) to service container
* ***php artisan make:provider MyServiceProvider***
* ***register()***
* ***boot()***  ---- called after all other services are loaded
* *$defer = true;* ----- registration is deferred

**FACADES**

* *Illuminate\Support\Facades\Facade*
* Base class uses ***\_callStatic()*** magic-method
* is a class that provides object from container
* ***STATIC*** interface to classes in service container
* Static proxies
* More flexibility than traditional static

**CONTRACTS**

* Set of ***interfaces***
* Since interfaces defines compulsory their dependencies
* Interfaces gives succint , small , brief description

**ROUTES**

* Web.php / api.php
* *Route::get(“/path/url” , function(){ });*
* ***Get post put patch delete options***
* *Route::****match****( [ ‘get’, ‘post’ ] , “path/url” , function(){......});*
* *Route::****any****( “path/url” , function(){......});*
* *Route::****except****( [ ‘get’, ‘post’ ] , “path/url” , function(){......});*
* CSRF FIELD (in Forms): ***csrf\_field();***
* *Route :: get ( “path/url/****{id}****” , function(****$id****) { …..} );*
* *Route :: get ( “path/url/****{id?}****” , function(****$id=”defaultvale”****) { …..} );*
* *Route :: get ( “path/url/****{id}****” , function(****$id****) { …..} )->****where(“id” , “[0-9]+” )****;*
* *Route :: get ( “path/url” , function() { …..} )->****name(“MyRoute”)****;*
* *Route::****group****([“****prefix****” => “my/site” , “****namespace****” => “Admin”, “****middleware****” => “auth” ], function(){*

*Route 1…..*

*Route 2…..*

*});*

* Route-Model binding

*Route::get(“path/url/{id}” , function(****MyClass $myclass****){ …. });*

* FORM METHOD SPOOFING
  + **{{ method\_field(‘put’) }}**
* *Route::****current()****;*

**MIDDLEWARES**

*- app/Http/Middleware*

- *php artisan make:middleware MyMiddleware*

- ***$middleware*** ---- Global middlewares

- **$routeMiddleare** ----- route specific

- ***web*** and ***api*** middleware groups

**CSRF PROTECTION**

* Cross-site request forgery
* ***{{ csrf\_field() }}***
* If vue.js and no auth scaffholding then add

<script>

window.Laravel = {!!

json\_encode([

“csrfToken” => csrf\_token( ) ;

])

!!} ;

</script>

* ***X-CSRF (AJAX)***

*<meta name=”csrf-token” content=”{{* ***csrf\_token()*** *}}” >*

* If Jquery

*$.ajaxSetup([*

*Headers : {*

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

*}*

*]);*

**CONTROLLERS**

* *app/Http/Controllers*
* *Route::get(“path/url” , “my\path\****AnyController@index“****)->middleware(‘auth’) ;*
* *php artisan make:controller MyController --****resource***
  + Route::**resource**(“/url” , “MyController” );
    - “only” => [ ‘index’ , ‘create’ ]
    - “Except” => [ “destroy”, “store” ]

**REQUESTS**

* *Illuminate\Http\Request*
  + *Typehint Eg : public function(****Request $request****) { ….. }*
* Input trimming & normalization --- is done by laravel
* ***$input = $request->all()***
* ***$input = $request->input(“name”);***
* ***$input = $request->name;***
* ***$input = $request->input(“name”, “defaultName”) ;***
* ***4input = $request->input(“products.0.name”); ------*** if arrays use ***dot***
* *$input = $request->****only****(“name”, “password”)*
* *$input = $request->****except****(“name”, “password”)*
* *$input = $request->****has****(“name”);*
* *$request->****flash****();* -------all inputs copy in session to next page/request
* flashOnly( [ “a” , ”b” ] );
* flashExcept( [ “a” , ”b” ] );
* *redirect(‘formpage’)->****withInput****() ;*
* *{{ old(“name”) }}*
* *COOKIE*
  + $request->cookie(“name”) ;
  + *Return response(“hello”)->cookie(“key”, “value”, $minutes)*
* *FILES*
  + *$request->****file****(“photo”) ;*
  + *$request->photo;*
  + *$request->****hasFile****(“photo”);*
  + *$request->file(“photo”)->****isValid****();*
  + *$request->file(“photo”)->extension();*
  + *$request->file(“photo”)->****store****(“path”);*
  + *$request->file(“photo”)->****store****(“path”,”s3”);*
  + *$request->file(“photo”)->****storeAs****(“path”, “filename.jpg”, “s3”);*

**RESPONSES**

* ***return*** *“hello”;*
* *return [ 3, 22, 76 ]* ---- is converted to JSON by laravel and returned
* ***response****(“hello” , 200 )->****header****(“content-type” , “text/plain”)->****header****(“othercontent”, “othervalue” );*
* *reponses($content)->withHeaders([*

*“H1” => “v1”,*

*“H2” => “v2”,*

*])*

* ***back****()->withInput()*
* ***redirect(“url”)***
* *redirect()->****route****(“routeName”)*
* *redirect()->****route****(“routeName”, [ “id” => 5 ] )*
* *redirect()->****actions****(“MycController@index”)*
* *redirect()->****actions****(“MycController@index”, [“id” =>4])*
* *redirect(“url”)->****with****(“myKey” , “myValue”);*
  + Access as {{ ***session****(“myKey”) }}*
* *redirect()->****view****(“hello” , $data, 200)->****header****(“H1”, “V1”) ;*
* *response()->****json****( [ “name” => “u”, “age” => 26 ]*
* *response()->****download****(“path” , “name.jpg” , $headerOptional)*
* *response()->****download****(“path” , “name.jpg” )->****deleteFileAfterSend****()*
* *response()->****file****(“path”)*

**VIEWS**

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

**SESSIONS**

* file/cookie/database/redis/memcached
* Database
  + Needs a table with schema
  + *php artisan session:table*
* Redis
  + Install ***predis/predis*** package
  + *config/database.php*
  + *config/session.php*
* *$request->****session****()->get(“key”)*
* *$request->****session****()->get(“key”, “default”)*
* ***session****(“key”)* -----gettting
* ***session****(“key” , “default”)*
* ***session****( [ “key” => “value” ])*
* *session()->****all()***
* *session()->****exists****(“key”)*
* *session()->****put****(“key”, “value”)*
* *session()->****pull****(“key” , “default”)*
* *session()->****flash****(“status” , “task was successful”)*
* *session(“)->****forget****(“key”)*
* *session(“)->****flush****();*
* *session(“)->****regenrate****()*

**VALIDATION**

* *$this->****validate****($request , [*

*“title” => “required|max:255” ,*

*“age” : “required|nullable” ] );*

* ***$errors***
* *php artisan make:request StoreBlogPost*

**BLADE**

* In layout *app.blade.php*
  + ***@yield(‘title’)***
* In extending file
  + ***@extends(****‘layout.app’)*
  + ***@section(‘title’)*** *…….@****endsection****()*
* *{{ $anyvar }}*
* *@****if*** *(count($a) == 5)*

*Hi*

*@****elseif*** *(count($a) == 4)*

*Good*

*@****else***

*Bye*

*@endif*

* @**for** ($a = 1 ; $a < 6 ; $a ++ )

Hiiii

@endfor

* @**foreach**($a as $k)

Hiii

@endforeach

* *@****while*** *($a < 5)*

*Hiii*

*@endwhile*

* ***$loop->index***
* ***$loop->iteration***
* ***$loop->first***
* ***@php***

*$a = 5;*

***@endphp***

* ***@include****(‘app.header’ , [ ‘key’ => ‘value’ ] )*

**JS/CSS SCAFFHOLDING**

* *package.json*
* ***npm install***
* ***npm run dev*** ------ each time
* ***npm run watch*** ------ continuous
* *Webpack.mix.js ---->* instructions
* ***assets****/sass/app.css|js ----> raw*
* ***public****/css|js -----> compiled / minified etc*
* Vue components
  + *assets/js/components/ Example.vue*
  + *Example.vue Is a single file vue component ---- html + js*
  + ***Vue component (****‘example****’ , require(‘****./components/Example.vue****’) )***

**LARAVEL MIX**

* *Install* ***nodejs*** *which has* ***npm***
* *npm install -----> package.json*
* Run mix in ***webpack.mix.js***
  + ***npm run dev***
  + ***Npm run production***
  + ***npm run watch***
* URL
  + TO STOP URL CHANGES BY MIX
  + mix.sass(‘asset/css/a.scss’ , ‘public/css/a.css’).options( [ processCssUrls: false ] )
* *If (mix.config.inProduction) {*

***mix.version();***

*}*

**AUTHENTICATION**

* *config/auth.php*
* ***Guards*** ----> web/api ----> how users are authenticated
* ***Providers*** ----->eloquent/query
* ***php artisan make:auth***
* ***$redirectTo = “/”*** -----> in login/register/resetpassword Controllers
* *Auth****::user();***
* *Auth****::id();***
* *Auth****::check()***
* Manual Authentication
  + *$a = Auth::****attempt****( [ ‘email’ => $email , ‘password’ => $password ] )*

*If ( $a ) { …..}*

* *Auth::****guard(****‘admin’)->attempt($a);*
* *Auth::****logout****();*
* *Auth::****login****($userObj)*
* *Auth::****guard****(“admin”)->login($userObj)*
* *Auth::****once****($credentials)* -------> for stateless auth in API

**API AUTH PASSPORT**

* PACKAGE: laravel/passport
* Register service provider as
  + *Laravel\Passport\PassportServiceProvider::class*
* *php artisan migrate*
* *Php artisan passport:install*
* *In* ***App\User*** *model Add trait Laravel\Passport\HasApiTokens*
* *In AuthServiceProvider add Passport::routes();*
* *In config/auth.php set driver as* ***“passport”*** *in api guard*

**AUTHORIZATION**

**ENCRYPTION**

* OpenSSL
* AES-256 / AES-128
* *php artisan key:generate -----> AES-256-CBC key*
* ***Crypt****::****encryptString*** *(‘hello’) ;*
* ***Crypt****::****decryptString*** *($encryptedKey) ;*

**HASHING**

* BCRYPT
* ***Hash****::****make****(“mypass”)*
* *Hash::****check****(“plainText” , $hashedPass )*

**RESET PASSWORD**

* *abc.com/password/reset*
* Reset token expires after 1 hour

**ARTISAN CONSOLE**

* ***php artisan***
* *list*
* *help migrate*
* *Tinker*
* *make:command MyCmd*
* *app/Console/Commands*

**CACHE**

* Memcached / Redis
* *Composer require* ***predis/predis***
* ***Cache****::****get****(“key”);*
* Access Cache stores
  + *Cache:: store( ‘file’ )->get( ‘foo’ );*
  + Stores array in *config/cache.php*
  + *Cache:: store(‘redis’)->put( ‘bar’ , ‘baz’ , 10 );*
* *Cache::* ***put****( ‘bar’ , ‘baz’ , 10 );*
* *Cache::* ***forget****(“key’)*
* *cache(‘key’)*

**COLLECTION**

* Uses array
* $collection = **collect**( [ 5, 2, 6 ] )
* $collection = **collect**( [ 5, 2, 6 ] )->**map**( function($name) {

return strtoupper($name) ;

}) ;

**ERRORS**

* *App\Exceptions\Handler*
* *Monolog library*
* ***abort(404);***

**EVENTS**

* Events ---> *app/Events*
* Listener ---> *app/Listeners*
* 1 event can have multiple listeners which do not depend on each other

**FILESYSTEM**

* 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);*

*}*