



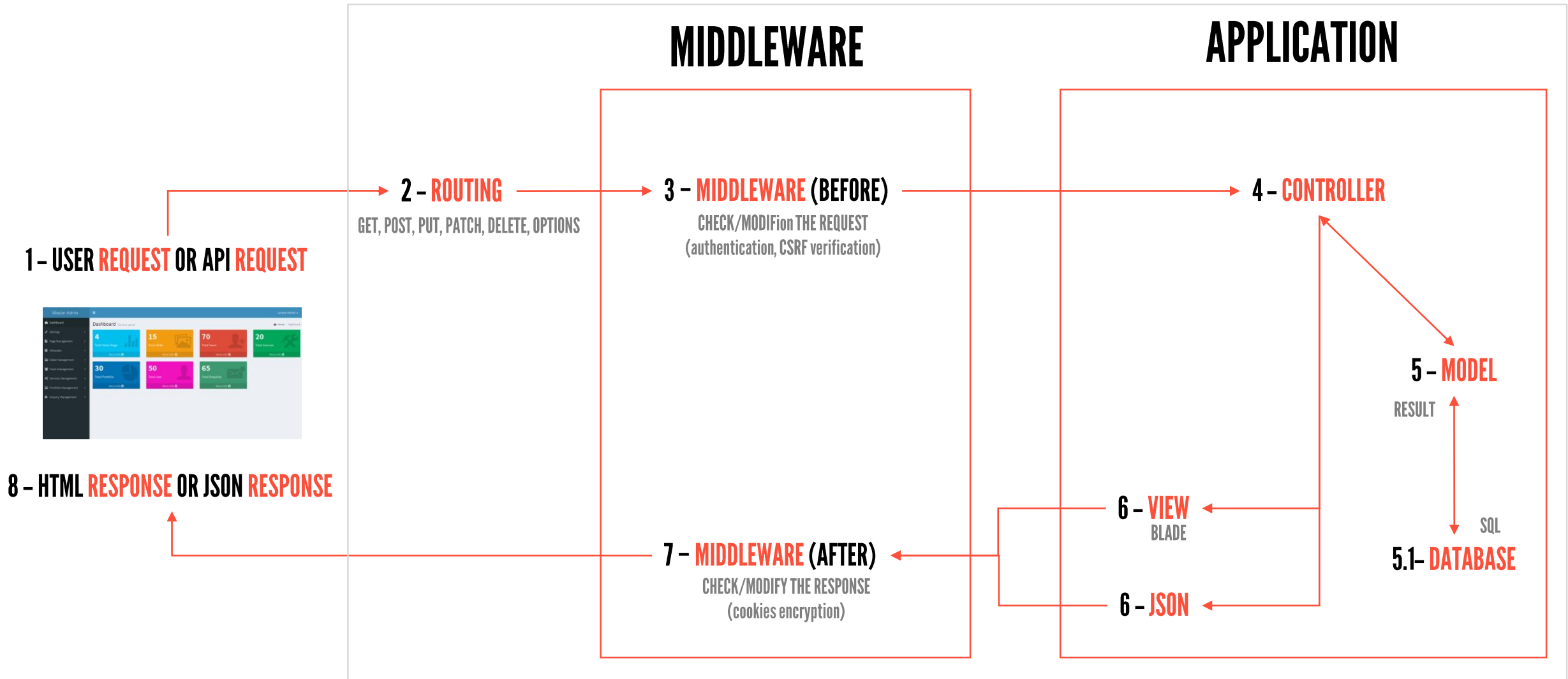
LARAVEL

BASICS

by dimitri.casier@howest.be

THE BIG PICTURE!

LARAVEL



A man with a surprised expression, wearing large black goggles and a white life preserver. He is in a dark, possibly underwater or night-time setting. The text "WHAT A LOVELY DAY" is overlaid in white, bold, sans-serif font with a black outline.

WHAT A LOVELY DAY

ARTISAN

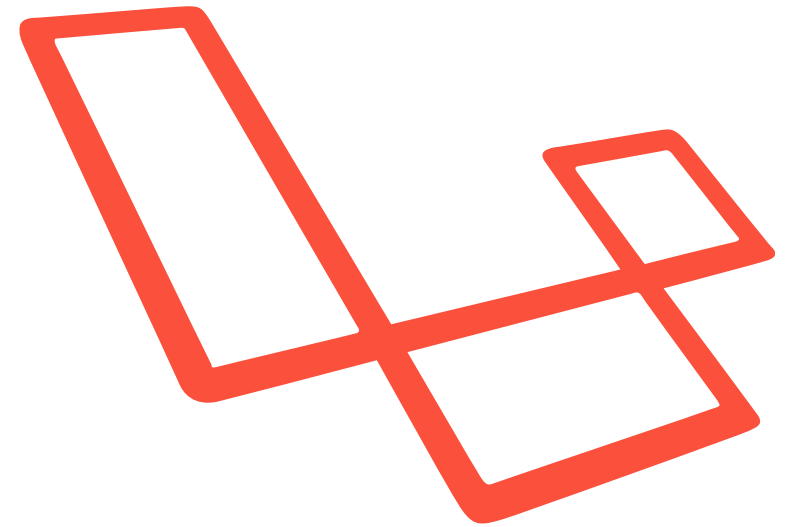
ROUTES

CONTROLLERS

VIEWS

MIGRATIONS

MODELS



START YOUR ENGINES!

HOMESTEAD VIRTUAL MACHINE

cd to Homestead folder
on local machine

vagrant up

vagrant ssh

cd code/<project-folder>

C:\vagrant@homestead: ~/code/laravel

```
C:\Users\dimitri.casier>cd Documents\Homestead
```

```
C:\Users\dimitri.casier\Documents\Homestead>vagrant up
```

```
Bringing machine 'homestead-7' up with 'virtualbox' provider...
```

```
==> homestead-7: Checking if box 'laravel/homestead' version '7.0.0' is up to date...
```

```
==> homestead-7: A newer version of the box 'laravel/homestead' for provider 'virtualbox' is
```

```
==> homestead-7: available! You currently have version '7.0.0'. The latest is version
```

```
==> homestead-7: '7.1.0'. Run `vagrant box update` to update.
```

```
==> homestead-7: Machine already provisioned. Run `vagrant provision` or use the `--provision`
```

```
==> homestead-7: flag to force provisioning. Provisioners marked to run always will still run.
```

```
C:\Users\dimitri.casier\Documents\Homestead>vagrant ssh
```

```
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-43-generic x86_64)
```

```
homestead
```

```
* Homestead 8.0.0 released! PHP 7.3 is now the default!
```

```
* Settler v7.0.0 released! Make sure you update
```

```
* Need PHP 5.6 or 7.0? Homestead 7.x Settler 6.4.0
```

```
0 packages can be updated.
```

```
0 updates are security updates.
```

```
Last login: Fri Feb 15 13:23:24 2019 from 10.0.2.2
```

```
vagrant@homestead:~$ dir
```

```
code
```

```
vagrant@homestead:~$ cd code/laravel/
```

```
vagrant@homestead:~/code/laravel$ dir
```

| | | | | | |
|-----------|---------------|--------------|-----------|------------|----------------|
| app | composer.json | database | public | routes | tests |
| artisan | composer.lock | package.json | readme.md | server.php | vendor |
| bootstrap | config | phpunit.xml | resources | storage | webpack.mix.js |

```
vagrant@homestead:~/code/laravel$
```

howest.be

we develop people

ARTISAN

ARTISAN THE LARAVEL COMMAND LINE INTERFACE

php artisan list

The list of commands

php artisan list make

The list of commands for make

php artisan help migrate

Every command has a help screen

A FEW NAMESPACES:

make:*

creates a new Laravel file/class/command/...

cache:*

commands for the cache

migrate:*

commands for migrations

vagrant@homestead: ~/code/laravel

command [options] [arguments]

Options:

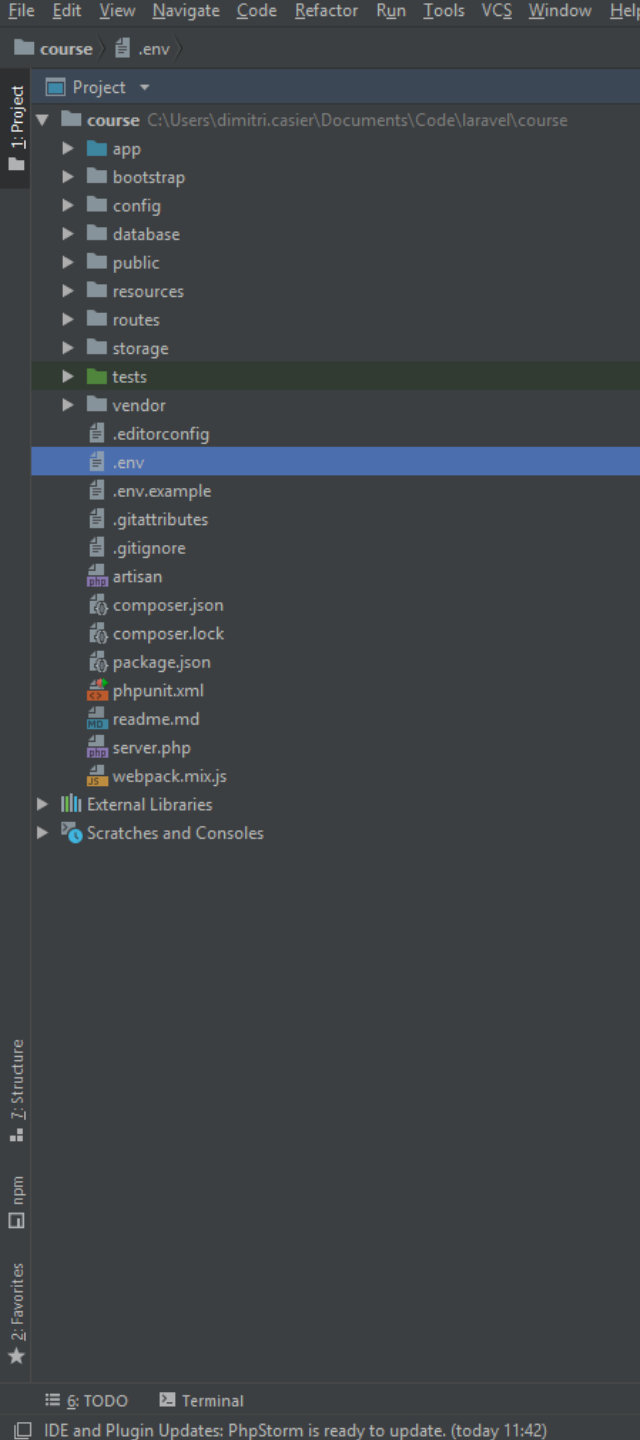
| | |
|----------------------|--|
| -h, --help | Display this help message |
| -q, --quiet | Do not output any message |
| -V, --version | Display this application version |
| --ansi | Force ANSI output |
| --no-ansi | Disable ANSI output |
| -n, --no-interaction | Do not ask any interactive question |
| --env[=ENV] | The environment the command should run under |
| -v vv vvv, --verbose | Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug |

Available commands for the "make" namespace:

| | |
|-------------------|--|
| make:auth | Scaffold basic login and registration views and routes |
| make:channel | Create a new channel class |
| make:command | Create a new Artisan command |
| make:controller | Create a new controller class |
| make:event | Create a new event class |
| make:exception | Create a new custom exception class |
| make:factory | Create a new model factory |
| make:job | Create a new job class |
| make:listener | Create a new event listener class |
| make:mail | Create a new email class |
| make:middleware | Create a new middleware class |
| make:migration | Create a new migration file |
| make:model | Create a new Eloquent model class |
| make:notification | Create a new notification class |
| make:observer | Create a new observer class |
| make:policy | Create a new policy class |
| make:provider | Create a new service provider class |
| make:request | Create a new form request class |
| make:resource | Create a new resource |
| make:rule | Create a new validation rule |
| make:seeder | Create a new seeder class |
| make:test | Create a new test class |

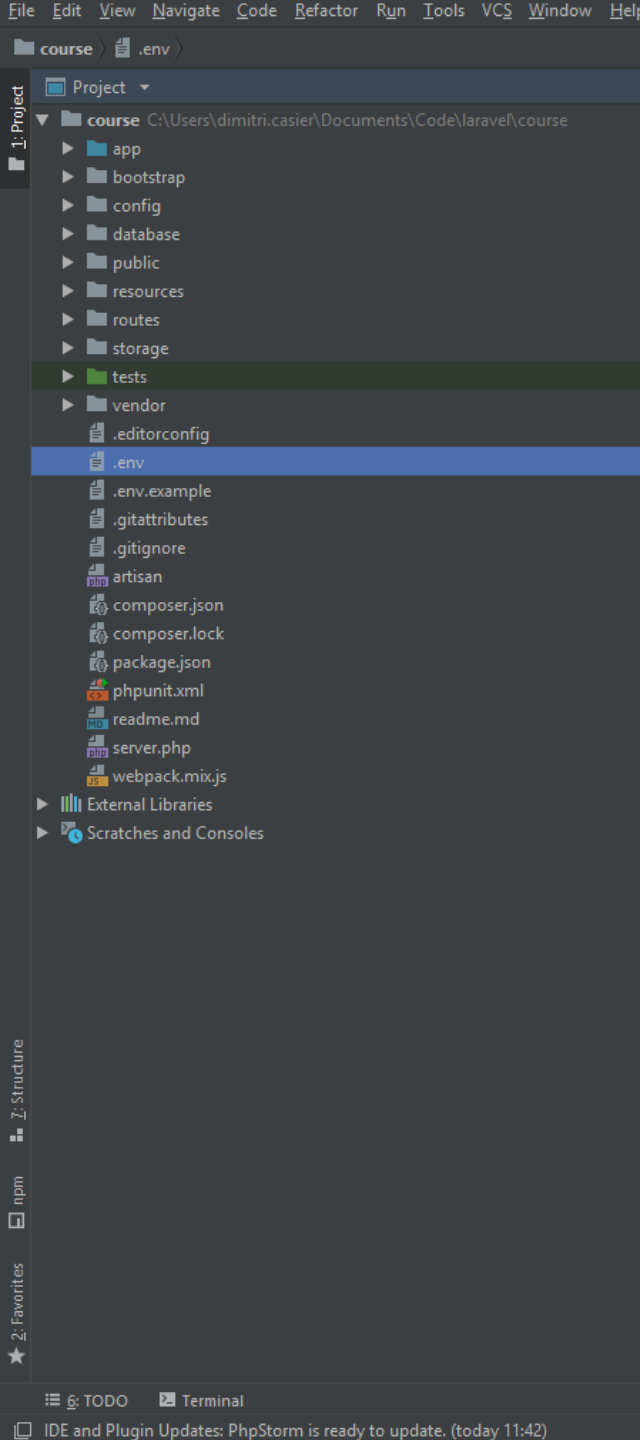
vagrant@homestead:~/code/laravel\$

DIRECTORY STRUCTURE



ROOT DIRECTORY

| | |
|------------------------|--|
| .env: | storing configuration, never commit this! (se gitignore) |
| .gitignore: | ignore me and commit me!! |
| artisan: | artisan command application |
| composer.json: | php package manager config file |
| composer.lock: | create by composer lock dependencies to versions |
| package.json: | front-end package manager config file |
| webpack.mix.js: | wrapper around webpack, to make your life easier |



ROOT DIRECTORY

vendor:

packages installed with composer + autoload logic

test:

all test files

storage:

upload files, logs, cache, sessions, compiled views

routes:

web, api, console

resources:

views and front-end development files

public:

public folder with index.php

database:

migrations, seeds and factories

config:

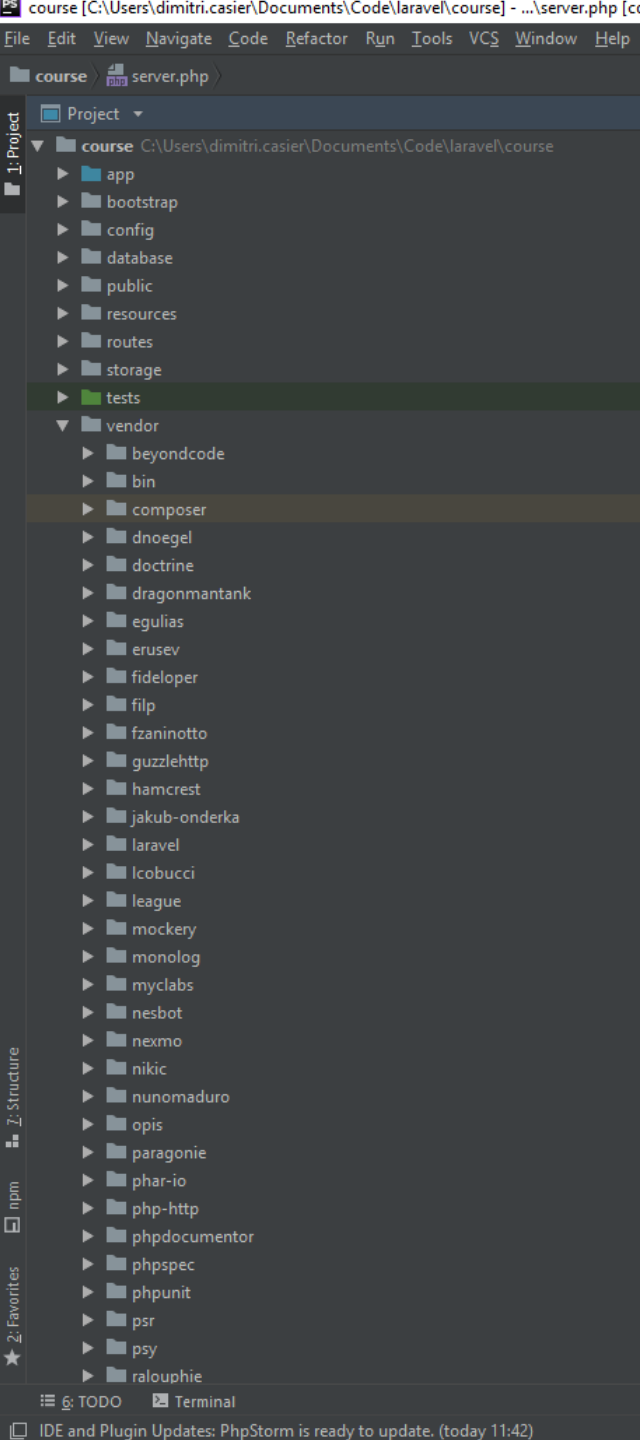
all config files laravel and/or packages

bootstrap:

app.php: bootstrap file of laravel

app:

console, controllers, middleware, providers and models



COMPOSER AND VENDOR DIRECTORY

Containing the packages installed with composer.

Never ever edit the files in the vendor folder!

Containing the **autoload** logic and files

<https://packagist.org/>

composer install
composer update (<package-name>)
composer require <package-name>

```
{
  "license": "MIT",
  "require": {
    "php": "^7.1.3",
    "fideloper/proxy": "^4.0",
    "laravel/framework": "5.7.*",
    "laravel/tinker": "^1.0"
  },
  "require-dev": {
    "beyondcode/laravel-dump-server": "^1.0",
    "filp/whoops": "^2.0",
    "fzaninotto/faker": "^1.4",
    "mockery/mockery": "^1.0",
    "nunomaduro/collision": "^2.0",
    "phpunit/phpunit": "^7.0"
  },
  "config": {
    "optimize-autoloader": true,
    "preferred-install": "dist",
    "sort-packages": true
  },
  "extra": {
    "laravel": {
      "dont-discover": []
    }
  },
  "autoload": {
    "psr-4": {
      "App\\": "app/"
    }
  },
  "autoload-dev": {
    "psr-4": {
      "Tests\\": "tests/"
    }
  }
}
```

/routes

ROUTE METHODS

```
41  
42 Route::get($uri, $callback);  
43 Route::post($uri, $callback);  
44 Route::put($uri, $callback);  
45 Route::patch($uri, $callback);  
46 Route::delete($uri, $callback);  
47 Route::options($uri, $callback);
```

RESPONDS TO MULTIPLE METHODS

```
49  
50 Route::match(['get', 'post'], '/', function () {  
51     //  
52 });  
53  
54 Route::any('/test', function () {  
55     //  
56 });  
57
```

SIMPLE ROUTE

```
14
15 Route::get('/', function () {
16     return view( view: 'welcome' );
17 });
18
```

SIMPLE NAMED ROUTE


```
14
15 Route::get('/', function () {
16     return view( view: 'welcome' );
17     ->name( 'welcome' );
18
19
```

URLS TO NAMED ROUTES

```
30
31 // -- url
32 $url = route( name: 'welcome' );
33
34 // redirect
35 return redirect() -> route( route: 'home' );
36
```


ROUTE PARAMETERS

```
59
60 Route::get('user/{id}', function ($id) {
61     return 'User '.$id;
62 })->name('user.show');
63
```

 Install Dependencies

```
65
66 return redirect()->route(route: 'user.show', ['id' => 1]);
67
```

OPTIONAL ROUTE PARAMETERS

```
65 Route::get('user/{name?}', function ($name = null) {
66     return $name;
67 })->name('user.show.name');
68
69 Route::get('user/{name?}', function ($name = 'John') {
70     return $name;
71 })->name('user.show.name');
72
```

ROUTE CONSTRAINTS (REGULAR EXPRESSIONS)

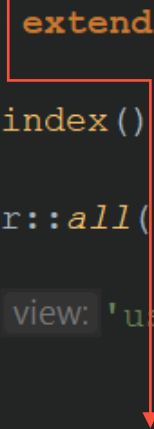
```
77
78 Route::get('user/{name}', function ($name) {
79     //
80     })->where('name', '[A-Za-z]+');
81
82 Route::get('user/{id}', function ($id) {
83     //
84     })->where('id', '[0-9]+');
85
86 Route::get('user/{id}/{name}', function ($id, $name) {
87     //
88     })->where(['id' => '[0-9]+', 'name' => '[a-z]+']);
89
```

ROUTE, CONTROLLERS AND PARAMETERS

```
20
21 Route::get('/users', 'UserController@index')->name('users.index');
22 Route::get('user/{id}', 'UserController@show')->name('users.show');
23
```

CONTROLLER AND PARAMETERS

```
6 class UserController extends Controller
7 {
8     public function index(){
9
10         $users = User::all();
11
12         return view( view: 'users.index', ['users' => $users]);
13     }
14
15     public function show($id){
16         $user = User::where('id', $id)->first();
17
18         return view( view: 'users.show', ['user' => $user]);
19     }
20 }
```



ROUTE MIDDLEWARE


```
21
22 Route::get('/home', 'HomeController@index')->name('home');
23 Route::get('/users', 'UserController@index')->name('user.index')->middleware('auth');
24 Route::get('user/{id}', 'UserController@show')->name('user.show')->middleware('auth');
```

ROUTE GROUP MIDDLEWARE

```
21
22 Route::get('/home', 'HomeController@index')->name('home');
23
24 Route::middleware(['auth'])->group(function() {
25     Route::get('/users', 'UserController@index')->name('user.index');
26     Route::get('user/{id}', 'UserController@show')->name('user.show');
27 });
```

ROUTE NESTED GROUP AND ROUTE NAME PREFIXES

```
21
22 Route::get('/home', 'HomeController@index')->name('home');
23
24 Route::middleware(['auth'])->group(function() {
25     Route::name('user.')->group(function () {
26         Route::get('/users', 'UserController@index')->name('index');
27         Route::get('user/{id}', 'UserController@show')->name('show');
28     });
29 });
```



```
21
22 Route::get('/home', 'HomeController@index')->name('home');
23
24 Route::middleware(['auth'])->group(function() {
25     Route::get('/users', 'UserController@index')->name('user.index');
26     Route::get('user/{id}', 'UserController@show')->name('user.show');
27 });
```

FALLBACK ROUTE

```
32  
33 - Route::fallback(function () {  
34     return 'not found fallback';  
35 - });  
36
```

ROUTES AND FORMS

```
38
39 // -- PUT, PATCH AND DELETE not supported on html forms
40 <form action="/foo/bar" method="POST">
41     <input type="hidden" name="_method" value="PUT">
42     <input type="hidden" name="_token" value="{{ csrf_token() }}">
43 </form>
44
45
46 // OR
47
48 <form action="/foo/bar" method="POST">
49     @method('PUT')
50     @csrf
51 </form>
52
```

/app/Http/Controllers

BASIC CONTROLLER

```
1  <?php
2  namespace App\Http\Controllers;
3
4  use App\User;
5  use Illuminate\Http\Request;
6
7  class UserController extends Controller
8  {
9      public function index(){
10
11          $users = User::all();
12
13          return view('users.index', ['users' => $users]);
14      }
15
16      public function show($id){
17
18          $user = User::where('id', $id)->first();
19
20          return view('users.show', ['user' => $user]);
21      }
22  }
```

Database

CONTROLLERS AND MIDDLEWARE

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class HomeController extends Controller
8  {
9      /**
10       * Create a new controller instance.
11       *
12       * @return void
13       */
14     public function __construct()
15     {
16         $this->middleware( middleware: 'auth' );
17
18         // -- or ->except('all');
19         $this->middleware( middleware: 'log' )->only( methods: 'index' );
20     }
21
22     /**
23      * Show the application dashboard.
24      *
25      * @return \Illuminate\Contracts\Support\Renderable
26      */
27     public function index()
28     {
29         return view( view: 'home' );
30     }
31 }
```

RESOURCE CONTROLLERS

```
vagrant@homestead:~/code/course$ php artisan make:controller --help
Description:
  Create a new controller class

Usage:
  make:controller [options] [--] <name>

Arguments:
  name                The name of the class

Options:
  -m, --model[=MODEL]  Generate a resource controller for the given model.
  -r, --resource        Generate a resource controller class.
  -i, --invokable       Generate a single method, invokable controller class.
  -p, --parent[=PARENT] Generate a nested resource controller class.
  --api                Exclude the create and edit methods from the controller.
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
  --ansi               Force ANSI output
  --no-ansi            Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
  --env[=ENV]          The environment the command should run under
  -v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
vagrant@homestead:~/code/course$
```

RESOURCE CONTROLLERS

```
1 namespace App\Http\Controllers;
2
3 use Illuminate\Http\Request;
4
5 class UserResourceController extends Controller
6 {
7     /**
8      * Display a listing of the resource.
9      *
10     * @return \Illuminate\Http\Response
11     */
12     public function index()
13     {
14         //
15     }
16
17     /** Show the form for creating a new resource. ... */
18     public function create() {...}
19
20     /**
21     * Store a newly created resource in storage.
22     *
23     * @param \Illuminate\Http\Request $request
24     * @return \Illuminate\Http\Response
25     */
26     public function store(Request $request) {...}
27
28     /**
29     * Display the specified resource.
30     *
31     * @param int $id
32     * @return \Illuminate\Http\Response
33     */
34     public function show($id) {...}
35
36     /**
37     * Show the form for editing the specified resource.
38     */
39 }
```

```
20
21 Route::resource('users', 'UserResourceController');
22
23 Route::resource('users', 'UserResourceController')->only(['index', 'show']);
24 Route::resource('users', 'UserResourceController')->except(['create', 'store', 'update', 'destroy']);
25
```


DEPENDENCY INJECTION: CONSTRUCTOR INJECTION AND METHOD INJECTION

```
1  <?php
2  namespace App\Http\Controllers;
3
4  use App\User;
5
6  class UserController extends Controller
7  {
8      private $model = null;
9
10     // -- constructor injection
11     public function __construct(User $model){
12         $this->model = $model;
13     }
14
15     // -- method injection
16     public function index(User $model){
17
18         $users = $model->all();
19
20         return view('users.index', ['users' => $users]);
21     }
22
23     public function show($id){
24
25         $user = $this->model->where('id', $id)->first();
26
27         return view('users.show', ['user' => $user]);
28     }
29 }
```

DEPENDENCY INJECTION

AND

SERVICE CONTAINERS



We'll talk about this later, okay?

/resources/views

VIEWS AND PARAMETERS

```
16 public function index(User $model){  
17  
18     $users = $model->all();  
19  
20     // return view('users.index')->with('users', $users);  
21     return view( view: 'users.index', ['users' => $users] );  
22 }  
23
```



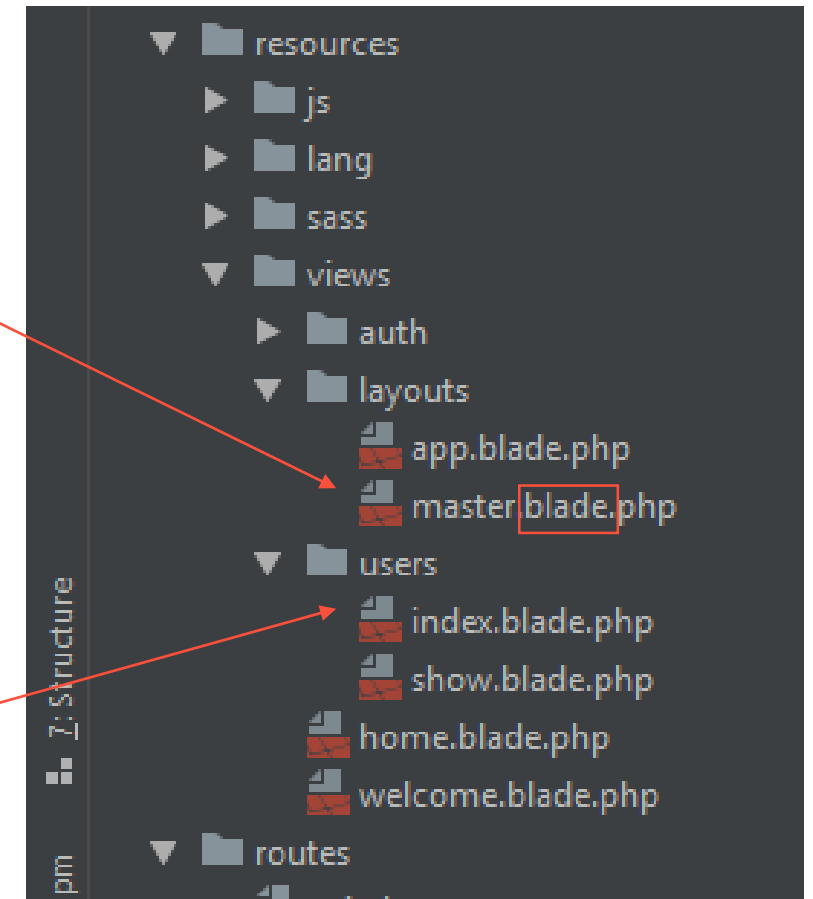
```
4 <section>  
5     <ul>  
6         @foreach($users as $user)  
7             <li>  
8                 {{ $user->id }} - {{ $user->name }} - {{ $user->email }}  
9             </li>  
10        @endforeach  
11    </ul>  
12 </section>
```

BLADE: MASTER VIEW (layouts.master)

```
1 <html>
2 <head>
3   <title>App Name - @yield('title', 'no title')</title>
4 </head>
5 <body>
6   @section('navigation')
7     navigation
8   @show
9
10  <div class="container">
11    @yield('content')
12  </div>
13 </body>
14 </html>
```

BLADE: DETAIL VIEW (users.index)

```
1 @extends('layouts.master')
2
3 @section('navigation')
4   @parent
5   sub navigation
6 @endsection
7
8 @section('content')
9   <section>
10     <ul>
11       @foreach($users as $user)
12         <li>
13           {{ $user->id }} - {{ $user->name }} - {{ $user->email }}
14         </li>
15       @endforeach
16     </ul>
17   </section>
18 @endsection
```



BLADE: RENDERING

```
{{ $user->name }} // escaped by htmlspecialchars  
  
{!! $user->name !!} // -- as html  
  
@json($user) // -- json_encode($user)  
  
Name, @{{ name }} // -- escapes the {{{}}
```

BLADE: INCLUDE

```
@include('menus.main', ['items' => $items])
```

BLADE: IF

```
29
30 @if (count($users) === 1)
31     I found one user!
32 @elseif (count($users) > 1)
33     I found multiple users!
34 @else
35     I found no users!
36 @endif
37
```

BLADE: LOOPS

```
39 @for ($i = 0; $i < 10; $i++)
40     The current value is {{ $i }}
41 @endfor
42
43 @foreach ($users as $user)
44     <p>This is user {{ $user->id }}</p>
45 @endforeach
46
47 @while (true)
48     <p>I'm looping forever.</p>
49 @endwhile
50
```

<https://laravel.com/docs/5.7/blade#control-structures>



[HTTPS://LARAVEL.COM/DOCS/5.7/BLADE](https://laravel.com/docs/5.7/blade)

/database/migrations

NEED A DATABASE?

.ENV

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:B8WxpQ2YbO0BZXK/MTZBiYY1ITjb7JTgoAEQlJ/pUDI=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=course
13 DB_USERNAME=homestead
14 DB_PASSWORD=secret
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
18 QUEUE_CONNECTION=sync
19 SESSION_DRIVER=file
20 SESSION_LIFETIME=120
21
22 REDIS_HOST=127.0.0.1
23 REDIS_PASSWORD=null
24 REDIS_PORT=6379
25
26 MAIL_DRIVER=smtp
27 MAIL_HOST=smtp.mailtrap.io
28 MAIL_PORT=2525
29 MAIL_USERNAME=null
30 MAIL_PASSWORD=null
31 MAIL_ENCRYPTION=null
32
33 PUSHER_APP_ID=
34 PUSHER_APP_KEY=
35 PUSHER_APP_SECRET=
```


DATABASE.CONFIG

```
<?php

return [

    /*...*/

    'default' => env( key: 'DB_CONNECTION', default: 'mysql' ),

    /*...*/

    'connections' => [

        'sqlite' => [
            'driver' => 'sqlite',
            'database' => env( key: 'DB_DATABASE', database_path( path: '
            'prefix' => '',
            'foreign_key_constraints' => env( key: 'DB_FOREIGN_KEYS',

        ],

        'mysql' => [
            'driver' => 'mysql',
            'host' => env( key: 'DB_HOST', default: '127.0.0.1' ),
            'port' => env( key: 'DB_PORT', default: '3306' ),
            'database' => env( key: 'DB_DATABASE', default: 'forge' ),
            'username' => env( key: 'DB_USERNAME', default: 'forge' ),
            'password' => env( key: 'DB_PASSWORD', default: '' ),
            'unix_socket' => env( key: 'DB_SOCKET', default: '' ),
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'prefix_indexes' => true,
            'strict' => true,
            'engine' => 'InnoDB',
            'table_prefix' => ''
        ]
    ]
];
```

?

×

Instellingen

Geavanceerd

Statistieken

Netwerktype

MariaDB or MySQL (TCP/IP)

Hostnaam / IP

127.0.0.1

☐ Vraag om inloggegevens

☐ Gebruik Windows authenticatie

Gebruiker

homestead

Wachtwoord

●●●●●●

Poort

33060

☐ Gecomprimeerd client/server protocol

Databases:

Punkomma gescheiden


Commentaar:

Openen

Annuleren

Meer

DATABASE MANAGER



Hey, I'm just trying to help!

OK...,

MIGRATIONS!

MIGRATIONS: **A VERSION CONTROL** **FOR DATABASES**



CREATE OR UPDATE A MIGRATION FILE

CREATE TABLE COMMANDS:

```
php artisan make:migration create_users_table
```

```
php artisan make:migration create_users_table --create=tbl_users
```

UPDATE TABLE COMMAND:

```
php artisan make:migration add_votes_to_users_table --table=users
```

MIGRATION COMMANDS:



```
vagrant@homestead:~/code/course$ php artisan list migrate
Laravel Framework 5.7.26
```

Usage:

```
command [options] [arguments]
```

Options:

| | |
|----------------------|--|
| -h, --help | Display this help message |
| -q, --quiet | Do not output any message |
| -V, --version | Display this application version |
| --ansi | Force ANSI output |
| --no-ansi | Disable ANSI output |
| -n, --no-interaction | Do not ask any interactive question |
| --env[=ENV] | The environment the command should run under |
| -v vv vvv, --verbose | Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug |

Available commands for the "migrate" namespace:

| | |
|------------------|---|
| migrate:fresh | Drop all tables and re-run all migrations |
| migrate:install | Create the migration repository |
| migrate:refresh | Reset and re-run all migrations |
| migrate:reset | Rollback all database migrations |
| migrate:rollback | Rollback the last database migration |
| migrate:status | Show the status of each migration |

```
vagrant@homestead:~/code/course$
```

MIGRATION STRUCTURE

UP()

ADD OR UPDATE TABLE BY ADDING, UPDATING
AND/OR REMOVING COLUMNS OR INDEXES

DOWN()

REVERSE THE UP() OPERATION

```
class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'users', function (Blueprint $table) {
            $table->increments( column: 'id' );
            $table->string( column: 'name' );
            $table->string( column: 'email' )->unique();
            $table->timestamp( column: 'email_verified_at' )->nullable();
            $table->string( column: 'password' );
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists( table: 'users' );
    }
}
```


CREATING COLUMNS



AND THAT'S HOW YOU DO IT

```
$table->bigIncrements('id'); // -- Auto-incrementing UNSIGNED BIGINT (primary key) equivalent column.
$table->bigInteger('votes'); // -- BIGINT equivalent column.
$table->binary('data'); // -- BLOB equivalent column.
$table->boolean('confirmed'); // -- BOOLEAN equivalent column.
$table->char('name', 100); // -- CHAR equivalent column with an optional length.
$table->date('created_at'); // -- DATE equivalent column.
$table->dateTime('created_at'); // -- DATETIME equivalent column.
$table->dateTimeTz('created_at'); // -- DATETIME (with timezone) equivalent column.
$table->decimal('amount', 8, 2); // -- DECIMAL equivalent column with a precision (total digits) and scale (decimal digits).
$table->double('amount', 8, 2); // -- DOUBLE equivalent column with a precision (total digits) and scale (decimal digits).
$table->enum('level', ['easy', 'hard']); // -- ENUM equivalent column.
$table->float('amount', 8, 2); // -- FLOAT equivalent column with a precision (total digits) and scale (decimal digits).
$table->geometry('positions'); // -- GEOMETRY equivalent column.
$table->geometryCollection('positions'); // -- GEOMETRYCOLLECTION equivalent column.
$table->increments('id'); // -- Auto-incrementing UNSIGNED INTEGER (primary key) equivalent column.
$table->integer('votes'); // -- INTEGER equivalent column.
$table->ipAddress('visitor'); // -- IP address equivalent column.
$table->json('options'); // -- JSON equivalent column.
$table->jsonb('options'); // -- JSONB equivalent column.
$table->lineString('positions'); // -- LINESTRING equivalent column.
$table->longText('description'); // -- LONGTEXT equivalent column.
$table->macAddress('device'); // -- MAC address equivalent column.
$table->mediumIncrements('id'); // -- Auto-incrementing UNSIGNED MEDIUMINT (primary key) equivalent column.
$table->mediumInteger('votes'); // -- MEDIUMINT equivalent column.
$table->mediumText('description'); // -- MEDIUMTEXT equivalent column.
$table->morphs('taggable'); // -- Adds taggable_id UNSIGNED BIGINT and taggable_type VARCHAR equivalent columns.
$table->multilineString('positions'); // -- MULTILINESTRING equivalent column.
$table->multiPoint('positions'); // -- MULTIPOINT equivalent column.
$table->multiPolygon('positions'); // -- MULTIPOLYGON equivalent column.
$table->nullableMorphs('taggable'); // -- Adds nullable versions of morphs() columns.
$table->nullableTimestamps(); // -- Alias of timestamps() method.
$table->point('position'); // -- POINT equivalent column.
$table->polygon('positions'); // -- POLYGON equivalent column.
$table->rememberToken(); // -- Adds a nullable remember_token VARCHAR(100) equivalent column.
$table->smallIncrements('id'); // -- Auto-incrementing UNSIGNED SMALLINT (primary key) equivalent column.
$table->smallInteger('votes'); // -- SMALLINT equivalent column.
$table->softDeletes(); // -- Adds a nullable deleted_at TIMESTAMP equivalent column for soft deletes.
$table->softDeletesTz(); // -- Adds a nullable deleted_at TIMESTAMP (with timezone) equivalent column for soft deletes.
$table->string('name', 100); // -- VARCHAR equivalent column with an optional length.
$table->text('description'); // -- TEXT equivalent column.
$table->time('sunrise'); // -- TIME equivalent column.
$table->timeTz('sunrise'); // -- TIME (with timezone) equivalent column.
$table->timestamp('added_on'); // -- TIMESTAMP equivalent column.
$table->timestampTz('added_on'); // -- TIMESTAMP (with timezone) equivalent column.
$table->timestamps(); // -- Adds nullable created_at and updated_at TIMESTAMP equivalent columns.
$table->timestampsTz(); // -- Adds nullable created_at and updated_at TIMESTAMP (with timezone) equivalent columns.
$table->tinyIncrements('id'); // -- Auto-incrementing UNSIGNED TINYINT (primary key) equivalent column.
$table->tinyInteger('votes'); // -- TINYINT equivalent column.
$table->unsignedBigInteger('votes'); // -- UNSIGNED BIGINT equivalent column.
$table->unsignedDecimal('amount', 8, 2); // -- UNSIGNED DECIMAL equivalent column with a precision (total digits) and scale (decimal digits).
$table->unsignedInteger('votes'); // -- UNSIGNED INTEGER equivalent column.
$table->unsignedMediumInteger('votes'); // -- UNSIGNED MEDIUMINT equivalent column.
$table->unsignedSmallInteger('votes'); // -- UNSIGNED SMALLINT equivalent column.
$table->unsignedTinyInteger('votes'); // -- UNSIGNED TINYINT equivalent column.
$table->uuid('id'); // -- UUID equivalent column.
$table->year('birth_year'); // -- YEAR equivalent column.
```

RENAMING / DROPPING TABLE

```
Schema::rename('tbl_users', 'users');

Schema::drop( table: 'users' );
Schema::dropIfExists( table: 'users' );
```

MODIFYING COLUMNS

```
Schema::table( table: 'users', function (Blueprint $table) {
    // -- examples
    $table->string( column: 'name', length: 50 )->change();
    $table->renameColumn( 'name', 'user_name' );
    $table->dropColumn( columns: 'name' );
    $table->dropColumn( ['name', 'email' ] );
});
```

/app/<MODELS>



BACK TO REALITY

MODELS

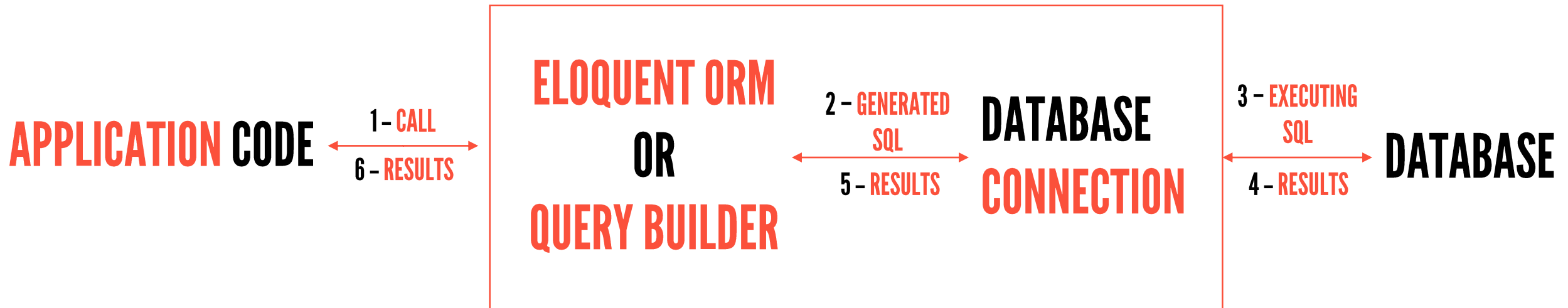
ELOQUENT ORM



ELOQUENT

OBJECT RELATION MAPPING

LARAVEL



ELOQUENT ORM

ACTIVE RECORD

A close-up of a man with a mustache and a confused expression, with his eyebrows furrowed and a slight frown. He is wearing a blue suit jacket, a white shirt, and a patterned tie. The background is a blurred office setting with wood paneling.

What did you say?

ACTIVE RECORD:
EACH TABLE HAS A

CORRESPONDING MODEL

USER EXTENDS MODEL

```
1 <?php
2
3 namespace App;
4
5 use Illuminate\Notifications\Notifiable;
6 use Illuminate\Contracts\Auth\MustVerifyEmail;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8
9 class User extends Authenticatable
10 {
11     use Notifiable;
12
13     /**
14      * The attributes that are mass assignable.
15      *
16      * @var array
17      */
18     protected $fillable = [...];
19
20     /**
21      * The attributes that should be hidden for arrays.
22      *
23      * @var array
24      */
25     protected $hidden = [...];
26
27 }
28
29
30
31
```

1

1

USERS TABEL

Basis

Opties

Indices

Refererende sleutels

Partities

CREA

Naam:

users

Commentaar:

Kolommen:

Toevoegen

Verwijder

Omhoog

Omlaag

| # | Naam | Datatype | Lengte/Set | Unsign... | NULL to... |
|---|-------------------|-----------|------------|-------------------------------------|-------------------------------------|
| 1 | id | INT | 10 | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| 2 | name | VARCHAR | 255 | <input type="checkbox"/> | <input type="checkbox"/> |
| 3 | email | VARCHAR | 255 | <input type="checkbox"/> | <input type="checkbox"/> |
| 4 | email_verified... | TIMESTAMP | | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 5 | password | VARCHAR | 255 | <input type="checkbox"/> | <input type="checkbox"/> |
| 6 | remember_to... | VARCHAR | 100 | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 7 | created_at | TIMESTAMP | | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 8 | updated_at | TIMESTAMP | | <input type="checkbox"/> | <input checked="" type="checkbox"/> |

Help

Annuleren

Opslaan



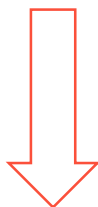
CREATE A MODEL FILE

CREATE MODEL COMMANDS:

`php artisan make:model Product`

`php artisan make:model Product -migration`

`php artisan make:model Product -m`



```
namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    //
}
```

```
vagrant@homestead:~/code/course$ php artisan make:model --help
Description:
  Create a new Eloquent model class

Usage:
  make:model [options] [--] <name>

Arguments:
  name                        The name of the class

Options:
  -a, --all                  Generate a migration, factory, and resource controller for the model
  -c, --controller          Create a new controller for the model
  -f, --factory              Create a new factory for the model
  --force                    Create the class even if the model already exists
  -m, --migration            Create a new migration file for the model
  -p, --pivot                Indicates if the generated model should be a custom intermediate table model
  -r, --resource             Indicates if the generated controller should be a resource controller
  -h, --help                Display this help message
  -q, --quiet                Do not output any message
  -V, --version              Display this application version
  --ansi                     Force ANSI output
  --no-ansi                  Disable ANSI output
  -n, --no-interaction       Do not ask any interactive question
  --env[=ENV]                The environment the command should run under
  -v|vv|vvv, --verbose       Increase the verbosity of messages: 1 for normal output, 2 for more
                             verbose output and 3 for debug
vagrant@homestead:~/code/course$
```

FETCHING

```
// -- returns a collection of models
$users = App\User::all();
$users = App\User::where('id', '>', 100)->orderBy('email')->get();
$users = App\User::find([100,101,102]);

// -- returns a model instance
$user = App\User::find(100);
$user = App\User::where('id', '>', 100)->first();

// -- returns a scalar value
$count = App\User::where('id', '>', 100)->count();
```

FETCHING IN CHUNKS

```
User::where('id', '>', 100)->chunk(200, function ($users) {
    foreach ($users as $user) {
        //
    }
});
```


PAGINATE

```
Route::get('/products', function() {  
    return \App\Product::paginate(2);  
});
```

```
{  
  "current_page": 1,  
  "data": [  
    {  
      "id": 1,  
      "name": "New name",  
      "created_at": "2019-02-28 08:12:43",  
      "updated_at": "2019-02-28 09:05:13"  
    },  
    {  
      "id": 2,  
      "name": "New name",  
      "created_at": "2019-02-28 08:13:08",  
      "updated_at": "2019-02-28 09:06:33"  
    }  
  ],  
  "first_page_url": "http://course.test/products?page=1",  
  "from": 1,  
  "last_page": 3,  
  "last_page_url": "http://course.test/products?page=3",  
  "next_page_url": "http://course.test/products?page=2",  
  "path": "http://course.test/products",  
  "per_page": 2,  
  "prev_page_url": null,  
  "to": 2,  
  "total": 6  
}
```

```
Route::get('/products', function() {  
    return \App\Product::where('id', '>', 2)->paginate(2);  
});
```

```
{  
  "current_page": 2,  
  "data": [  
    {  
      "id": 5,  
      "name": "no name",  
      "created_at": "2019-02-28 08:43:28",  
      "updated_at": "2019-02-28 09:32:00"  
    },  
    {  
      "id": 6,  
      "name": "no name",  
      "created_at": "2019-02-28 08:56:41",  
      "updated_at": "2019-02-28 09:32:00"  
    }  
  ],  
  "first_page_url": "http://course.test/products?page=1",  
  "from": 3,  
  "last_page": 2,  
  "last_page_url": "http://course.test/products?page=2",  
  "next_page_url": null,  
  "path": "http://course.test/products",  
  "per_page": 2,  
  "prev_page_url": "http://course.test/products?page=1",  
  "to": 4,  
  "total": 4  
}
```

INSERTING

```
Route::post('/products', function(\Illuminate\Http\Request $request, \App\Product $model){  
  
    // -- after saving, the model contains values for  
    // -- id, created_at, updated_at and name  
  
    $model->name = $request->get(key: 'name', default: 'No name');  
    $model->save();  
  
    return $model->id;  
});
```

UPDATING

```
Route::put('/products/{id}', function(\App\Product $model, $id) {  
  
    $model = $model->find($id);  
    if (is_null($model)) {  
        return response([  
            'status' => 'error',  
            'message' => 'product not found!',  
        ], status: 404);  
    }  
    //endif  
  
    $model->name = 'New name';  
    $model->save();  
  
    return [  
        'status' => 'ok',  
        'message' => 'successfully updated',  
        'data' => $model  
    ];  
});
```

Install Dependencies

UPDATING MULTIPLE RECORDS

```
Route::post('/products/update', function(\App\Product $model) {  
  
    $nrOfUpdates = $model->where('id', '>', 2)  
        ->update([  
            'name' => 'no name'  
        ]);  
  
    return $nrOfUpdates;  
  
});
```

DELETING

```
// -- delete()  
$product = \App\Product::find(1);  
$product->delete();  
  
$deletedRows = Product::where('active', 0)->delete();  
  
// -- destroy  
Product::destroy(1,2,3);  
Product::destroy([1,2,3]);
```

FACADE

VS

NO FACADE

```
<?php
namespace App\Http\Controllers;

use App\Product;

class ProductController extends Controller
{
    public function index() {
        return Product::all();
    }

    public function show($id) {
        $model = Product::find($id);

        return $model;
    }
}
```

```
<?php
namespace App\Http\Controllers;

use App\Product;

class ProductController extends Controller
{
    private $model = null;

    public function __construct(Product $model)
    {
        $this->model = $model;
    }

    public function index() {
        return $this->model->all();
    }

    public function show($id) {
        $model = $this->model->find($id);

        return $model;
    }
}
```

MASS—ASSIGNMENT

**A MASS-ASSIGNMENT VULNERABILITY OCCURS WHEN
A USER PASSES AN UNEXPECTED HTTP PARAMETER THROUGH A REQUEST,
AND THAT PARAMETER CHANGES A COLUMN IN YOUR DATABASE YOU DID NOT EXPECT.**

MASS-ASSIGNMENT CREATE

VS

NO MASS-ASSIGNMENT SAVE

```
// -- inserts
\App\Product::create(request()->all());

// -- updates
$product = \App\Product::find(1);
$product->fill(['name' => 'new name']);
$product->save();
```

```
$product = new Product;
$product->name = 'product 1';
$product->price = 100;
$product->save;
```

MODEL PROPERTIES

WELCOME!!!

GET OUT!!!

```
$product = App\Flight::create(['name' => 'product 1', 'price' => '100']);
```

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

// -- change table name, default is model_name + s

protected \$table = 'data_products';

// -- when primary key is not the default: id

protected \$primaryKey = 'my_id';

// -- when primary key is not a autonumbering

public \$incrementing = false;

// -- when primary key not an integer

protected \$keyType = 'string';

// -- not automatically manage by Eloquent

public \$timestamps = false;

// -- hidden for array

protected \$hidden = ['secret_sku'];

// -- mass assignable

protected \$fillable = ['name'];

// -- not mass assignable

protected \$guarded = ['price'];

}

UPDATING MULTIPLE RECORDS

```
Route::post('/products/update', function(\App\Product $model) {  
  
    $nrOfUpdates = $model->where('id', '>', 2)  
        ->update([  
            'name' => 'no name'  
        ]);  
  
    return $nrOfUpdates;  
  
});
```

DELETING

```
// -- delete()  
$product = \App\Product::find(1);  
$product->delete();  
  
$deletedRows = Product::where('active', 0)->delete();  
  
// -- destroy  
Product::destroy(1,2,3);  
Product::destroy([1,2,3]);
```

ELOQUENT RELATIONSHIPS = SELF STUDIE!
<https://laravel.com/docs/5.7/eloquent-relationships>



/database/seeds

CREATE A SEEDER (2 steps):

1. `php artisan make:seeder ProductSeeder`
2. `Composer dump-autoload`

You also can use **model factories** for creating data, see:

<https://laravel.com/docs/5.7/database-testing#writing-factories>

CALL / REGISTER A SEEDER:

`/database/seeds/DatabaseSeeder.php`

RUN A SEEDER:

`php artisan db:seed`

`php artisan db:seed -help` (for more info)

```
use Illuminate\Database\Seeder;

class ProductSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        for($i=0; $i<10; $i++)
        {
            \App\Product::create([
                'name' => 'product ' . Str::random(10)
            ]);
        }
    }
}
```

```
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        // $this->call(UsersTableSeeder::class);
        $this->call(class: ProductSeeder::class);
    }
}
```

TIME FOR A LITTLE



ARTISAN – EXERCISES:

1. Create a model Role.php, a controller RoleController.php and a migration file <date>_create_role_table.php with only one artisan command. Try to locate the model, the controller and the migration file.
2. What happens in the .env file if you run php artisan key:generate? What is this key? (Use only Laravel documentation!)
3. Show a list of all GET routes order by uri. How many routes in this list are using a Closure?
4. Run the list of artisan commands below and analyze what happens:
 - php artisan migrate:status
 - php artisan migrate
 - php artisan migrate
 - php artisan migrate:status
 - php artisan migrate:rollback
 - php artisan migrate:status
 - php artisan migrate:rollback
 - php artisan migrate
 - php artisan migrate:status

ROUTING – EXERCISES:

1. Create a route that calculates the Fibonacci sequence. Make use of a controller. The name of the route is fibonacci.number.

Example url: /fibonacci/10

2. Create a route /fibonacci that shows a list of links “Fibonacci of x” with x is 1 to 15, in a view. The href of the links are the route from previous exercise. Make use of the route name to create to links.

Hint: <https://laravel.com/docs/5.7/blade#loops>

CONTROLLER – EXERCISES:

1. Create a resource controller and a model to fetch records from the migration table, with only one artisan command.
Only the index and the show route are available through the routes. Return for both methods the results in json. Use method inject in the controller.
2. The index and show may only be visible when your logged in. Use route name prefixes for your routes.

VIEW & BLADE – EXERCISES:

1. Create a master view, an index view (migration list) and a show view (migration detail). Use blade syntax.
2. Add on each page a menu to navigate to the home page and to the migrations page. Make use of an include for the menu.

MIGRATION – EXERCISES:

1. Create a migration file for a the table Roles.

Fields:

- id
- name
- description
- active
- created_at
- updated_at

2. Update the table roles:
 - add the fields: created_by and last_updated_by
 - rename the field: name to role

ELOQUENT MODEL – EXERCISES :

1. Create a form with a search field that uses a like query on the field email and name of the table users. Show the results (id, name, email) when you hit the submit/search button.

Hint: where, orwhere, like, %, csrf (forms), request() -> see Laravel documentation

2. Add an edit link, /user/{id}/edit, to each row in the list of previous exercise. When you click on the link, you see a form with the name and email of the user and a save button.
After clicking on the save button, the form will send an post request to the UserController. This controller will update the name and email address of the user. Don't mind validation (yet!)

