LARAVEL 11

Part 1: Introduction, Installation and Routes

WHAT IS LARAVEL?

- Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.

- Laravel offers a rich set of functionalities which incorporates the basic features of PHP frameworks like CodeIgniter, Yii and other programming languages like Ruby on Rails. Laravel has a very rich set of features which will boost the speed of web development.

- If you are familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It saves a lot time if you are planning to develop a website from scratch. Moreover, a website built in Laravel is secure and prevents several web attacks.

ADVANTAGES

- The web application becomes more scalable, owing to the Laravel framework.
- Considerable time is saved in designing the web application, since Laravel reuses the components from other framework in developing web application.
- It includes namespaces and interfaces, thus helps to organize and manage resources.

COMPOSER

Composer is a tool which includes all the dependencies and libraries. It allows a user to create a project with respect to the mentioned framework (for example, those used in Laravel installation). Third party libraries can be installed easily with help of composer.

ARTISAN CLI

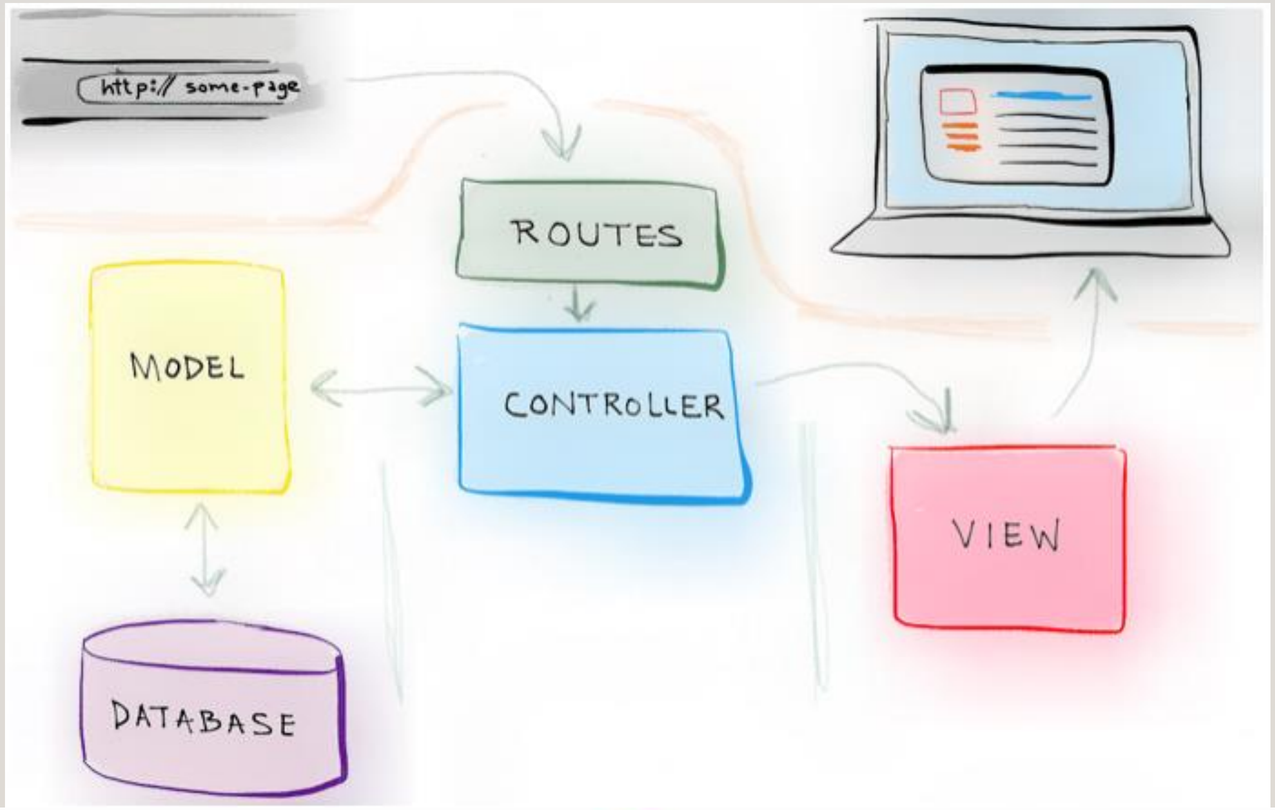
Command line interface used in Laravel is called Artisan. It includes a set of commands which assists in building a web application. These commands are incorporated from Symphony framework, resulting in add-on features in Laravel 5.1 (latest version of Laravel).

MVC

Model–view–controller (MVC) is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements. These elements are the internal representations of information (the Model), the interface (the View) that presents information to and accepts it from the user, and the Controller software linking the two.

Traditionally used for desktop graphical user interfaces (GUIs), this pattern became popular for designing web applications. Popular programming languages have MVC frameworks that facilitate the implementation of the pattern.

THE MVC STRUCTURE



LARAVEL TUTORIALS

https://laracasts.com/

INSTALL LARAVEL

- XAMPP , PHP 8.2 or higher
- Install Composer (Run as Administrator)
- Open CMD as Administrator and write below command to check if it installed correctly.

```
composer
```

- Go to Laravel.com and find the ways to install Laravel
- Go to htdocs to start your first Laravel application.

INSTALL LARAVEL

- Open CMD

```
composer global require laravel/installer
laravel new example-app
```

- Test Your Installation

TROUBLESHOOTING

- PHP zip extension
- Git not downloaded
- PHP version not compatible with the composer

LARAVEL FOLDERS STRUCTURES

- Http → controllers
- Models
- Resources → views
- vendor
- App
- Config
- database

LARAVEL FOLDERS STRUCTURES

- public
- routes
- storage
- tests
- .env file
- .gitignore file
- artisan file

GIT AND GITHUB

https://code.visualstudio.com/docs/sourcecontrol/intro-to-git

ROUTING

routes → web.php

ROUTING - GET

Return message

return ‘<h1> Welcome</h1>’;

Add path

```
Route::get('/welcome', function () {
    return 'First message on laravel';
});
```

ROUTING – SEND PARAMETER

```
Route::get('/car/{id}', function ($id) {
    return 'The id is: ' . $id;
});
```

ROUTING – SEND OPTIONAL PARAMETER

```
Route::get('/car/{id?}', function ($id = 0) {
    return 'The id is: ' . $id;
});
```

ROUTING – CONSTRAINTS
SEND PARAMETER WITH REGULAR EXPRESSION

```
Route::get('/car/{id?}', function ($id=0) {
    return 'The id is: ' . $id;
})->where(['id'=> '[0-9]+']);
```

ROUTING – CONSTRAINTS
SEND PARAMETER WITH WHERENUMBER

```
Route::get('/car/{id?}', function ($id=0) {
    return 'The id is: ' . $id;
})->whereNumber('id');
```

ROUTING – CONSTRAINTS
SEND PARAMETER WITH WHEREALPHA

```
Route::get('/car/{name?}', function ($name=null) {

    return 'The name is: ' . $name;

})->whereAlpha(‘name');
```

ROUTING – CONSTRAINT MULTIPLE PARAMETERS

```
Route::get('/car/{id?}/{name}', function ($id=0, $name) {
    return 'The id is: ' . $id . " " . $name;
})->where(['id'=>'[0-9]+', 'name'=>'[a-zA-Z]+']);
```

ROUTING – CONSTRAINTS WHEREIN

```
Route::get('/car/{name}', function ($name) {
    return "The name is: " . $name;
})->whereIn('name',['Peter', 'Tony']);
```

ROUTING – CONSTRAINTS PATTERNS APP → PROVIDERS

Go to RouteServiceProvider.php

Add the required constraints inside public function boot(): void

like this

```
public function boot(): void
    {
        Route::pattern('name','[a-zA-Z]+');
```

ROUTING – CONSTRAINTS

More details

https://laravel.com/docs/10.x/routing#parameters-regular-expression-constraints

ROUTE PREFIX

```
Route::prefix('cars')->group(function () {
    Route::get('bmw', function(){
        return 'BMW page';
    });

    Route::get('mercedes', function(){
        return 'Mercedes page';
    });
});
```

FALLBACK

```
Route::fallback(function() {
    return redirect('/');
});
```

Or

```
Route::fallback(fn() => redirect('/'));
```

ROUTE TO VIEW

Resources → views

Create file .blade.php inside the above path

Then use route as below

```
Route::get('test', function(){
    return view('car');
});
```

ROUTE TO VIEW

Or you can use

```
Route::view('test', 'car');
```

NAMED ROUTES

1- Create new blade file in views and add form code

```
<form action="{{ route('receive') }}" method="post">
     @csrf
     <input type="text" name="fullName">
     <input type="submit">
 </form>
```

NAMED ROUTES

2- inside the routes add below

```
Route::view('send', 'sendData');

Route::post('receive', function() {
    return 'data received';
})->name('receive');
```

MAKE CONTROLLER

Stop your server and write

```
php artisan make:controller controllerName
```

It will be created on

App → Http → Controllers

MAKE CONTROLLER

Inside the controller you can add

```
public function my_data(){
        return view('car');
    }
```

MAKE CONTROLLER

Then inside the routes web.php use the controller as below

```
Route::get('test', [exampleController::class, 'my_data']);
```

ROUTING WITH A FIRST CONTROLLER WITH NAMESPACE

- https://www.youtube.com/watch?v=75_t9Dxka1E&t=223s&ab_channel=PhpAnonymous

WHAT IS MIDDLEWARE IN LARAVEL?

https://laravel.com/docs/10.x/middleware

HOW MIDDLEWARE WORK?

app → Middleware

All middlewares connected with app → Http → Kernel.php

ROUTE WITH MIDDLEWARE

```
Route::middleware('auth')->get('data', fn()=>'Welcome’);


Route::middleware('guest')->get('data', fn()=>'Welcome’);


Route::get('data', fn()=>'Welcome’)->middleware('auth');
```

ROUTE WITH MIDDLEWARE AND CONTROLLER

To add middleware to whole controller methods, inside the controller we can add

```
public function __construct(){
        $this->middleware('auth');
    }
```

ROUTE WITH MIDDLEWARE AND CONTROLLER

Or add to a specific route from web.php routes as below

```
Route::get('test', [Controller::class,'car'])->middleware('auth');
```

CONTROLLER CRUD

CRUD means Creating, reading, updating, and deleting resources is used in pretty much every application. Laravel helps make the process easy using resource controllers. Resource Controllers can make life much easier and takes advantage of some cool Laravel routing techniques

From cmd

```
php artisan make:controller exampleController -r
```

KOICA Korea International Cooperation Agency

السويد Sverige

ECTD Egyptian Council For Training & Development

العمل اللائق للمرأة

منظمة العمل الدولية

هيئة الأمم المتحدة للمرأة

المجلس القومي للمرأة THE NATIONAL COUNCIL FOR WOMEN