

Part-1 : Laravel installation Process

I already had Laravel installed globally. So I used the following command in the project folder

```
laravel new TestProject1
```

```
user@MSI MINGW64 /g/xampp/htdocs/ostad/php-laravel/module-13
$ laravel new TestProject1

[ Laravel ]

Creating a "laravel/laravel" project at "./TestProject1"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v10.2.0)
 - Downloading laravel/laravel (v10.2.0)
   - Installing laravel/laravel (v10.2.0): Extracting archive
Created project in G:\xampp\htdocs\ostad\php-laravel\module-13\TestProject1
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 107 installs, 0 updates, 0 removals
 - Locking brick/math (0.11.0)
 - Locking dflydev/dot-access-data (v3.0.2)
 - Locking doctrine/inflexor (2.0.6)
 - Locking doctrine/lexer (3.0.0)
 - Locking dragonmantank/cron-expression (v3.3.2)
 - Locking egulias/email-validator (4.0.1)
 - Locking fakerphp/faker (v1.22.0)
 - Locking filp/whoops (2.15.2)
 - Locking fruitcake/php-cors (v1.2.0)
 - Locking graham-campbell/result-type (v1.1.1)
 - Locking guzzlehttp/guzzle (7.6.1)
 - Locking guzzlehttp/promises (1.5.2)
 - Locking guzzlehttp/psr7 (2.5.0)
 - Locking guzzlehttp/uri-template (v1.0.1)
 - Locking hamcrest/hamcrest-php (v2.0.1)
 - Locking laravel/framework (v10.10.1)
 - Locking laravel/pint (v1.10.0)
 - Locking laravel/sail (v1.22.0)
 - Locking laravel/sanctum (v3.2.5)
 - Locking laravel/serializable-closure (v1.3.0)
 - Locking laravel/tinker (v2.8.1)
 - Locking league/commonmark (2.4.0)
 - Locking league/config (v1.2.0)
 - Locking league/flysystem (3.15.1)
 - Locking league/flysystem-local (3.15.0)
 - Locking league/mime-type-detection (1.11.0)
 - Locking mockery/mockery (1.5.1)
 - Locking monolog/monolog (3.3.1)
 - Locking myclabs/deep-copy (1.11.1)
 - Locking nesbot/carbon (2.66.0)
 - Locking nette/schema (v1.2.3)
 - Locking nette/utils (v4.0.0)
 - Locking nikic/php-parser (v4.15.4)
 - Locking nunomaduro/collision (v7.5.2)
 - Locking nunomaduro/termwind (v1.15.1)
 - Locking phar-io/manifest (2.0.3)
 - Locking phar-io/version (3.2.1)
 - Locking phpoption/phpoption (1.9.1)
 - Locking phpunit/php-code-coverage (10.1.1)
 - Locking phpunit/php-file-iterator (4.0.2)
 - Locking phpunit/php-invoker (4.0.0)
 - Locking phpunit/php-text-template (3.0.0)
 - Locking phpunit/php-timer (6.0.0)
 - Locking phpunit/phpunit (10.1.3)
```

```

- Installing sebastian/erf-parser (1.0.0): Extracting archive
- Installing phpunit/php-timer (6.0.0): Extracting archive
- Installing phpunit/php-text-template (3.0.0): Extracting archive
- Installing phpunit/php-invoker (4.0.0): Extracting archive
- Installing phpunit/php-file-iterator (4.0.2): Extracting archive
- Installing theseer/tokenizer (1.2.1): Extracting archive
- Installing sebastian/lines-of-code (2.0.0): Extracting archive
- Installing sebastian/complexity (3.0.0): Extracting archive
- Installing sebastian/code-unit-reverse-lookup (3.0.0): Extracting archive
- Installing phpunit/php-code-coverage (10.1.1): Extracting archive
- Installing phar-io/version (3.2.1): Extracting archive
- Installing phar-io/manifest (2.0.3): Extracting archive
- Installing myclabs/deep-copy (1.11.1): Extracting archive
- Installing phpunit/phpunit (10.1.3): Extracting archive
- Installing spatie/backtrace (1.4.0): Extracting archive
- Installing spatie/flare-client-php (1.3.6): Extracting archive
- Installing spatie/ignition (1.7.0): Extracting archive
- Installing spatie/laravel-ignition (2.1.2): Extracting archive
 71 package suggestions were added by new dependencies, use `composer suggest`
to see details.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoLoadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
spatie/laravel-ignition ..... DONE

 81 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

 INFO  No publishable resources for tag [laravel-assets].

  No security vulnerability advisories found
> @php artisan key:generate --ansi

 INFO  Application key set successfully.

 INFO  Application ready! Build something amazing.

user@MSI MINGW64 /g/xampp/htdocs/ostad/php-laravel/module-13

```

Installation is done.

Validating the Installation:

```
MINGW64:/g/xampp/htdocs/ostad/php-laravel/module-13/TestProject1

user@MSI MINGW64 /g/xampp/htdocs/ostad/php-laravel/module-13
$ php artisan serve
Could not open input file: artisan

user@MSI MINGW64 /g/xampp/htdocs/ostad/php-laravel/module-13
$ cd TestProject1

user@MSI MINGW64 /g/xampp/htdocs/ostad/php-laravel/module-13/TestProject1
$ php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

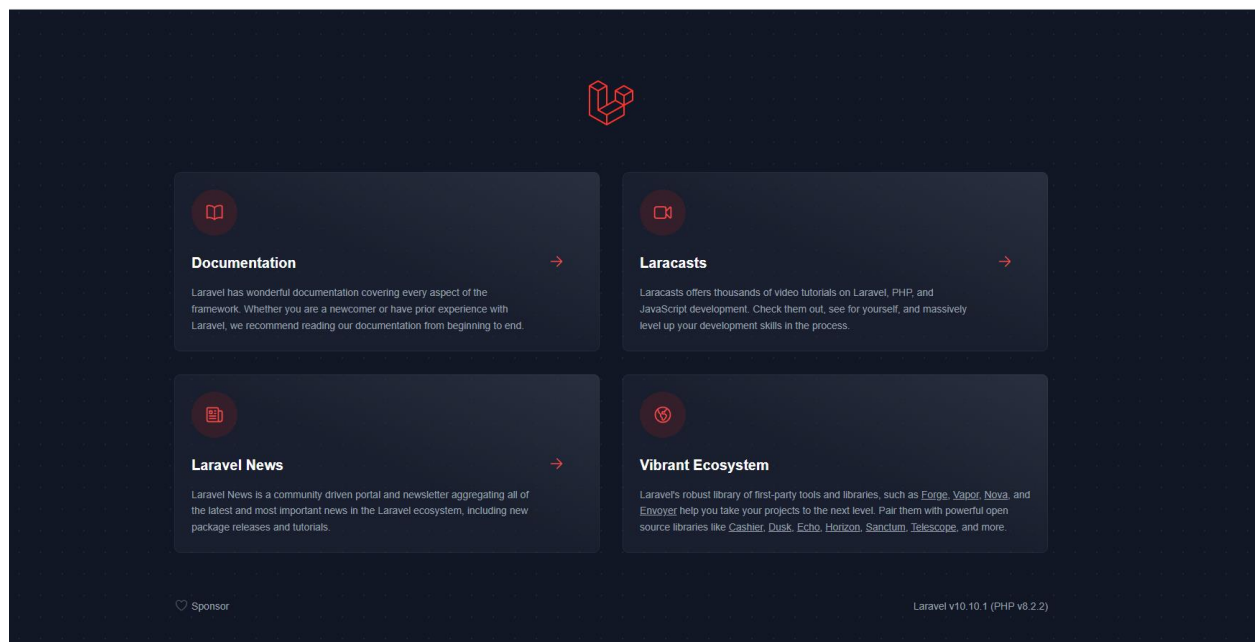
Press Ctrl+C to stop the server

user@MSI MINGW64 /g/xampp/htdocs/ostad/php-laravel/module-13/TestProject1
$ php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server

2023-05-16 19:26:20 .....
2023-05-16 19:26:41 /favicon.ico .....
```



Part-2: Laravel folder Structure

app: This directory holds the essential code of the application, encompassing models, controllers, and other classes that define the business logic and functionality.

bootstrap: The bootstrap directory contains the initialization files for the framework and its components. It consists of files responsible for autoloading, error handling, and configuration loading.

config: The config directory stores configuration files for various aspects of the Laravel application, such as database settings, mail configurations, service providers, and more. These files can be customized to modify the behavior of your application.

database: This directory manages the database-related files of the application. It includes migrations, which handle the creation and modification of database tables, and seeders, which allow you to populate the database with initial data.

public: The public directory serves as the web root of your Laravel application. It contains the entry point for incoming HTTP requests, the index.php file, as well as publicly accessible assets like CSS, JavaScript, and image files.

resources: The resources directory holds various resource files used by the application, such as views, language files, and assets that require compilation,

such as SCSS or JavaScript files. Views are stored in the views subdirectory and are responsible for rendering HTML templates.

routes: The routes directory contains route definition files that determine how the application responds to different HTTP requests. It includes web.php for handling web requests and api.php for handling API requests. In these files, you can define routes, middleware, and route groups.

storage: The storage directory is used to store temporary and long-term files generated by the application. This includes logs, session files, cache files, and uploaded files. Additionally, it serves as the default location for Laravel's file-based caching and session storage.

tests: The tests directory comprises automated tests for the application. It contains test cases that cover different parts of the application's functionality. Laravel provides a testing framework that facilitates the writing and execution of tests.

vendor: The vendor directory is managed by Composer, a dependency management tool used by Laravel. It contains all the third-party libraries and packages that your application depends on. Composer handles the installation and updates of these dependencies based on the requirements defined in the composer.json file.

Creating New Route:

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HelloController;

/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Route::get('/hello' , function (){
    return 'Hello World';
});

Route::get('/hc', [HelloController::class,'greet'] );
```

← → ↻ ⓘ 127.0.0.1:8000/hello

Hello World

Hello from controller

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HelloController extends Controller
{
    function greet(){
        return 'Hello From Controller!';
    }
    function greetAPI(){
        return 'Hello API';
    }
}
```

← → ↻ ⓘ 127.0.0.1:8000/hc

Hello From Controller!

API:

← → ↻ ⓘ 127.0.0.1:8000/api/helloapi

Hello API