

Laravel Framework

Report

# 

# 

# Laravel Framework

It is a very popular and secure MVC framework used for web development for full featured applications. MVC is a design pattern which means Model, View and Controller.

Model contains the application data and business logic whereas View displays the application view. In between the model and view is controller, which controls the request/response of users.

**Model:**

perform data validations, process data and store it

**Data from:**

flat file

database

XML document

Other valid data sources.

**Controller:**

Deals with the users’ requests for resources from the server.

between model and view

when the users requests for the URL …/index.php?products=list, the controller will load the products model to retrieve the products data

then output the results in the list view.

**View:**

Usually in the form of html pages.

It is a good, simple and elegant toolkit for web development.

# Why Laravel?

* It uses MVC architecture which in result secures the application by separating the view and model parts.
* It’s syntax makes it object oriented.
* Secure authentication and authorization.
* Rich support of libraries and components.
* It uses a dependency manager - Composer which manages the information to manage its packages
* Multiple file support system, also supports cloud storage such as amazon and rack space. It helps in making a distributed system such that we can use files from local storage as well as cloud storage in one application.
* It has its own CLI - Artisan, and we can create our customer commands by enhancing and extending the functionalities of Artisan.
* Eloquent ORM - Object Relational Mapper allows us to interact with database objects and relations.
* Blade Template Engine can combine multiple templates with data model to produce resulting views.
* Different folders and subfolders for broadcasting and events handling
* Broadcasting implements real time data and shows live feeds.
* It has multiple packages from different sources.
* Easy testing and laravel has its own unit testing.

**MVC Framework types:**

1. Code Igniter
2. Kohana
3. Cake PHP
4. Zend

**Prerequisites to learn Laravel:**

1. HTML
2. Core PHP
3. Advanced PHP

We can open application files with:

* Artisan Server
* XAMPP, WAMP servers

**2021 Best Frameworks:**

* Laravel
* Symfony
* CodeIgniter
* Zend Framework / Laminas Project
* Yii (Framework)
* CakePHP
* Slim
* Phalcon
* FuelPHP
* Fat-Free Framework

**Application Structure:**

It contains multiple folders for different functionality.

1. **App folder:**
   * Console subfolder: It contains 2 files, inspire.php and kernel.php. Commands are declared in inspire.php and are called from kernel.php
   * Events subfolder: It contains all the events of the application such as actions, validations, error messages etc
   * Exception subfolder: It contains all the methods for exception handling.
   * HTTP subfolder: It contains middleware, controller and view subfolders.
     + Middleware: Communication between request and response
     + Requests: Contains all the requests of the application.
   * Providers subfolder: It contains all the service providers files such as events provides, service provider etc
2. **Bootstrap:**

App.php contains the scripts which need to be initialized for bootstrap. Bootstrap file contains the bootstrap scripts.

It has subfolder cache which stores the files needed to run the application/website.

1. **Config:**

Contains all the configuration files for smooth functioning of application.

* **Database:**

It contains all the database needed files, and 3 sub directories.

* + Seeds: contains the classes for unit testing databases.
  + Migration: helps in querying the database migration.
  + Factories: Helps in creating a large number of data records.
* **Public:**

Root folder helps in initialization of application. The index.php file is stored in it which is opened as the first page of the web application.

* **Resources:**

It enhances the web application.

* + assets − It includes files that are required for styling the web application.
  + lang − It includes configuration for localization or internalization.
  + views − Views are the HTML files or templates which interact with end users and play a primary role in MVC architecture.
* **Storage:**

stores all the logs and necessary files of running laravel project

* + App: contains succession files.
  + Framework: contains sessions, caches and views which are frequently used.
  + Logs: Exceptions and error logs
* **Tests:**

Contains all the unit test cases for the application.

* **Vendor:**

Contains all the dependencies or laravel so that 3rd party applications and api can work

* **Githubignore**: contains files which are not to be committed.
* **Env:** contains all the credentials of users as well as database such as database username and table names etc

(It can not be committed to github/bitbucket thatswhy included in githubignore)

* **Env example**: contains default credentials as a backup
* **Composer.json** contains all the dependencies of laravel.

**api.php allow us to build APIs, we can make GET/POST/DELETE/PUT/PATCH to interact with other API or databases**

# 

# Routing

It works by mapping the URL requests with responses.

The request is sent through URL, controller loads the model to view, and page is shown as response.

2 things are required, URL and template/ page name.

**Syntax:**

Stored in web.php

We can store multiple routes in the file.

Route :: get(‘/’ , function ()

{

return view(‘welcome ’);

});

Get accepts the url requests and returns the response.

(Welcome is the html page → welcome.blade.php)

(/ is the route folder)

# Code to create a hello world Application:

**Simple but inconvenient:**

In web.php file:

Route:: get(‘/’ , function(){

return view(“Hello World”);

});

Writing /new instead of / can be accessed with localhost/laravel/new

**Passing Data with routing:**

**PRINT THE NAME TAKEN IN URL :**

|  |  |
| --- | --- |
| **Route: get (‘/ {name}’ , function($name){**  **echo $name;**  **return view(‘Welcome’);**  **});** | **Route: get (‘/{name}’ , function(‘$name){**  **Return view(‘Welcome’, [‘xyz’=> $name]);**  **});** |

**Proper Way:**

|  |  |
| --- | --- |
| **new.blade.php** | **web.php** |
| **<html>**  **<body>**  **<center> <h1> Hello World, I'm Shiwani </h1> </center>**  **</body>**  **</html>** | **Route:: get(‘/’ , function(){**  **return view(‘new’);**  **});** |

**Or simply output name by writing it in blade.php file**

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

**Anchor Tags:**

**Using a href tag to link pages.**

**< a href = ‘/about’ > About us </a>**

**About.blade.php exists so it will be opened.**