Full-Stack Web Development using Laravel with Vue.JS

Unit: I

Introduction to Laravel

Course Details
(B. Tech. 6th Sem)

Rajat Kumar
(Asst. Professor)
CSE Department

Brief Introduction of Faculty

COURSE	BOARD/UNIVERSITY	YEAR OF PASSING
M.Tech	ITM, AKTU, Lucknow	2020
B.Tech	ITM, AKTU, Lucknow	2016

- Certificate of Recognition for role as **Faculty** in MOOCs Course **Python/CSS** from **E & ICT Academy, IIT Kanpur**.
- Worked on Internet of things **(IoT)** technology under the topic **Energy efficient technique for data aggregation** in M.Tech.
- Published paper in International journal of research in electronics and computer engineering **(IJRECE)** on **"Energy efficient technique for data aggregation in IoT"** (April-June 2019).
- **Area of Interest**: FullStack Development, Machine Learning, IOT

Evaluation Scheme

NOIDA INSTITUTE OF ENGG. & TECHNOLOGY, GREATER NOIDA, GAUTAM BUDDH NAGAR
(AN AUTONOMOUS INSTITUTE)

Bachelor of Technology
Computer Science and Engineering
EVALUATION SCHEME
SEMESTER-VI

Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Scheme				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
1	ACSE0601	Advanced Java Programming	3	0	0	30	20	50		100		150	3
2	ACSE0602	Computer Networks	3	1	0	30	20	50		100		150	4
3	ACSE0603	Software Engineering	3	0	0	30	20	50		100		150	3
4		Departmental Elective -III	3	0	0	30	20	50		100		150	3
5		Departmental Elective -IV	3	0	0	30	20	50		100		150	3
6		Open Elective-I	3	0	0	30	20	50		100		150	3
7	ACSE0651	Advanced Java Programming Lab	0	0	2				25		25	50	1
8	ACSE0652	Computer Networks Lab	0	0	2				25		25	50	1
9	ACSE0653	Software Engineering Lab	0	0	2				25		25	50	1
10	ACSE0659	Mini Project	0	0	2				50			50	1
11	ANC0602 / ANC0601	Essence of Indian Traditional Knowledge / Constitution of India, Law and Engineering	2	0	0	30	20	50		50		100	
12		MOOCs (For B.Tech. Hons. Degree)											
		GRAND TOTAL										**1100**	**23**

Syllabus

UNIT-I: Introduction to Laravel

Introduction to Laravel, Laravel Features, Laravel installation, Application Structure of Laravel, Root Directory, App Directory, Basic Configuration, Environmental Configuration, Routing, Routing Parameters, Middleware, Terminable Middleware, Middleware Parameter, Controllers, Restful Resource Controllers, Implicit Controllers, Constructor Injection, Method Injection, Laravel Sail, Laravel Jetstream.

Syllabus

UNIT-II: Vue.js Framework&Inertia.js

Vue.js Template Syntax And Expressions, Vue directives, loops and conditional rendering, VueDevtools, Handling user Inputs, Handling Events, Vuejs Methods and Computed Properties, Attribute Bindings and dynamic classes, Concepts of Inertia.js, How it works, Inertia protocol, Routing, Responses and Pages, Creating links, GET, POST,PUT, PATCH, and DELETE method in Inertia.js

Syllabus

UNIT-III: Laravel Authentication&Laravel Faker

Laravel design patter, Laravel blade template engine, Artisan command, Login with username or email, Register with username or email, Logout, Validate request data (required, unique, etc..), Protecting Router, Password Confirmation, Social & Other Authentication method, Show success / Failure message, Faker PHP library, Create data seeder, Seed data, Localisation, Model Factories

Syllabus

UNIT-IV: Connecting Laravel with databases

Database Configuration File, Read/Write connections, Running A Select Query, Running an Insert, Update, Delete Statement, Listening For Query Events, Database Transaction, rollback and commit method, Accessing connections, Query Logging, Laravel Query Builder & ORM, Laravel Migration& Eloquent.

Syllabus

UNIT-V: Deployment Laravel application to production

PHPExtension:BCMath,Ctype,cURL,JSON,Mbstring,OpenSSL,PCRE, PDOServer Configuration, Nginx, Laravel server management service LaravelForge, Autoloader optimization, Optimizing Configuration Loading, Optimizing Route Loading, Optimizing View Loading, Debug Mode, Deploying With Vapor.

Branch Wise Application

Adaptability: It gives bother free migration, basic, and efficient structure.

Components: They help in creating custom elements that can be reused in HTML.

Transition: Various methodologies are given in Vue.js to apply a transition to HTML components when they are included or expelled from the DOM.

Detailed Documentation: It gives a simple learning curve through point-by-point documentation

Course Objective

1.Laravel Development Means Faster Time-To-Market. ...

2.Better Authentication and Authorization Option. ...

3.Stay Away from Technical Vulnerabilities with Laravel Framework. ...

4.MVC Architecture of Laravel Framework. ...

5.Automated and Unit Testing Feature. ...

6.Automated Task Execution and Scheduling

Course Outcomes (COs)

CO : 01.	**Apply the knowledge of PHP that are vital in understanding Laravel application and analyze the concepts, principles and methods in current Server-side technology to implement Laravel application over the web.**
CO : 02.	Explain, analyze and apply the role of Client-side scripting language like Vuejs in the workings of the web and web applications.
CO : 03.	Implementing and analyzing the concept of Larvel Faker and Authentication on Laravel.
CO : 04.	Understand the impact of web designing by database connectivity with different databases in the current market place where everyone use to prefer electronic medium for shoping, commerce, and even social life also.
CO : 05.	Analyzing and Creating a functional website using Laravel and Vuejs and Deploying and Optimizing Web Application using Forge / Vapor.

Program Outcomes (POs)

PO : 01.	Engineering Knowledge
PO : 02.	Problem Analysis
PO : 03.	Design/Development of solutions
PO : 04.	Conduct Investigations of complex problems
PO : 05.	Modern tool usage
PO : 06.	The engineer and society
PO : 07.	Environment and sustainability
PO : 08.	Ethics
PO : 09.	Individual and teamwork
PO : 10.	Communication
PO : 11.	Project management and finance
PO : 12.	Life-long learning

COs - POs Mapping

CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	2	3	3	-	-	-	-	-	-	-
CO2	3	2	3	2	3	-	-	-	-	-	-	-
CO3	3	2	3	2	3	-	-	-	-	-	-	-
CO4	3	2	3	2	3	-	-	-	-	-	-	-
CO5	3	2	3	3	3	-	-	-	-	-	-	-
AVG	2.8	2.0	2.8	2.4	3.0	-	-	-	-	-	-	-

Program Specific Outcomes(PSOs)

Sr. No.	PSO Description
PSO1	Understand to shows relationships and interactions between classes or objects of a pattern.
PSO2	Study to speed up the development process by providing well-tested, proven development
PSO3	Select a specific design pattern for the solution of a given design problem
PSO4	Create a catalogue entry for a simple design pattern whose purpose and application is understood.

COs - PSOs Mapping

CO.K	PSO1	PSO2	PSO3	PSO4
CO1	3	-	-	-
CO2	3	3	-	-
CO3	3	3	-	-
CO4	3	3	-	-
CO5	3	3	-	-

Program Educational Objectives (PEOs)

Sr. No.	PEOs Description
PEO 1	To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and provide sustainable solutions for real-life problems using state-of-the-art technologies.
PEO 2	To have a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors and to face the global challenges.
PEO3	To have an effective communication skills, professional attitude, ethical values and a desire to learn specific knowledge in emerging trends, technologies for research, innovation and product development and contribution to society.
PEO4	To have life-long learning for up-skilling and re-skilling for successful professional career as engineer, scientist, entrepreneur and bureaucrat for betterment of society.

Pattern of Online External Exam Question Paper (100 marks)

Printed page: .... **Subject Code: .....**

Roll No: | | | | | | | | | | | | |

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA

(An Autonomous Institute Affiliated to AKTU, Lucknow)

B.Tech./MBA/MCA/M.Tech (Integrated)

(SEM:....THEORY EXAMINATION(2020-2021)

Subject .....

Time: 2 Hours **Max. Marks: 100**

Pattern of Online External Exam Question Paper (100 marks)

		SECTION – A	**[30]**	**CO**
1.	**Attempt all parts- (MCQ, True False)Three Question From Each Unit**		**[15×2=30]**	
		UNIT-1		
	1-a.	**Question-**	**(2)**	
	1-b.	**Question-**	**(2)**	
	1-c.	**Question-**	**(2)**	
		UNIT-2		
	1-d.	**Question-**	**(2)**	
	1-e.	**Question-**	**(2)**	
	1-f.	**Question-**	**(2)**	
		UNIT-3		
	1-g.	**Question-**	**(2)**	
	1-h.	**Question-**	**(2)**	
	1-i.	**Question-**	**(2)**	
		UNIT-4		
	1-j.	**Question-**	**(2)**	
	1-k.	**Question-**	**(2)**	
	1-l.	**Question-**	**(2)**	
		UNIT-5		
	1-m.	**Question-**	**(2)**	
	1-n.	**Question-**	**(2)**	
	1-o.	**Question-**	**(2)**	

Pattern of Online External Exam Question Paper (100 marks)

		SECTION – B	**[20×2=40]**	**CO**
2.	**Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in Glossary)) Question from Unseen passage - Four Question From Unit-I**		**[4×2=08]**	**CO**
	Glossary- (Required words to be written)			
	2-a.	**Question-**	**(2)**	
	2-b.	**Question-**	**(2)**	
	2-c.	**Question-**	**(2)**	
	2-d.	**Question-**	**(2)**	
3.	**Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in Glossary) Question from Unseen passage - Four Question From Unit-II**		**[4×2=08]**	**CO**
		Glossary- (Required words to be written)		
	3-a.	**Question-**	**(2)**	
	3-b.	**Question-**	**(2)**	
	3-c.	**Question-**	**(2)**	
	3-d.	**Question-**	**(2)**	

Pattern of Online External Exam Question Paper (100 marks)

4.	Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in Glossary) Question from Unseen passage - Four Question From Unit-III		[4×2=08]	CO
		Glossary- (Required words to be written)		
	4-a.	Question-	(2)	
	4-b.	Question-	(2)	
	4-c.	Question-	(2)	
	4-d.	Question-	(2)	
5.	Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in Glossary) Question from Unseen passage - Four Question From Unit-IV		[4×2=08]	CO
		Glossary- (Required words to be written)		
	5-a.	Question-	(2)	
	5-b.	Question-	(2)	
	5-c.	Question-	(2)	
	5-d.	Question-	(2)	
6.	Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in Glossary) Question from Unseen passage - Four Question From Unit-V		[4×2=08]	CO
		Glossary- (Required words to be written)		
	6-a.	Question-	(2)	
	6-b.	Question-	(2)	
	6-c.	Question-	(2)	
	6-d.	Question-	(2)	

Pattern of Online External Exam Question Paper (100 marks)

		SECTION – C		
7	**Answer any10 out 15 of the following, Subjective Type Question, Three Question from Each Unit**		**[10×3=30]**	**CO**
		UNIT-1		
	7-a.	**-Question-**	**(3)**	
	7-b.	**-Question-**	**(3)**	
	7-c.	**-Question-**	**(3)**	
		UNIT-2		
	7-d.	**-Question-**	**(3)**	
	7-e.	**-Question-**	**(3)**	
	7-f.	**-Question-**	**(3)**	
		UNIT-3		
	7-g.	**-Question-**	**(3)**	
	7-h.	**-Question-**	**(3)**	
	7-i.	**-Question-**	**(3)**	
		UNIT-4		
	7-j.	**-Question-**	**(3)**	
	7-k.	**-Question-**	**(3)**	
	7-l.	**-Question-**	**(3)**	
		UNIT-5		
	7-m	**-Question-**	**(3)**	
	7-n.	**-Question-**	**(3)**	
	7-o.	**-Question-**	**(3)**	

Prerequisite and Recap

- Before begin progressing with this tutorial, we assume that we are familiar with website development using PHP and MySQL also we should be familiar with HTML, Core PHP, and Advance PHP. We have applied Laravel version 5.1 in all the examples.

Brief Introduction about the Subject with videos

YouTube /other Video Links:

- https://www.youtube.com/watch?v=1onmPIe07yo&ab_channel=Bitfumes
- https://www.youtube.com/watch?v=AhM4nAxaTLo&ab_channel=StellaLi
- https://www.youtube.com/watch?v=AEVhRhD2Wk&ab_channel=MattSocha
- https://www.youtube.com/watch?v=DfslJ-kaZXA&ab_channel=GeekyShows

Unit Content

- Introduction to Laravel, Laravel Features, Laravel installation, Application Structure of Laravel, Root Directory, App Directory, Basic Configuration, Environmental Configuration, Routing, Routing Parameters, Middleware, Terminable Middleware, Middleware Parameter, Controllers, Restful Resource Controllers, Implicit Controllers, Constructor Injection, Method Injection, Laravel Sail, Laravel Jetstream.

Unit Objective

- In this unit we will learn basic of Laravel.
- Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching.
- Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality.

Topic Objective

- Topic :Introduction to Laravel

In this topic, the students will learn about the basic of Laravel. And also basic of Laravel framework and MVC.

Introduction to Laravel

- **Laravel** is an open-source PHP framework. It also offers the rich set of functionalities that incorporates the basic features of PHP frameworks such as CodeIgniter, and other programming languages like Ruby on Rails.

What is Laravel?

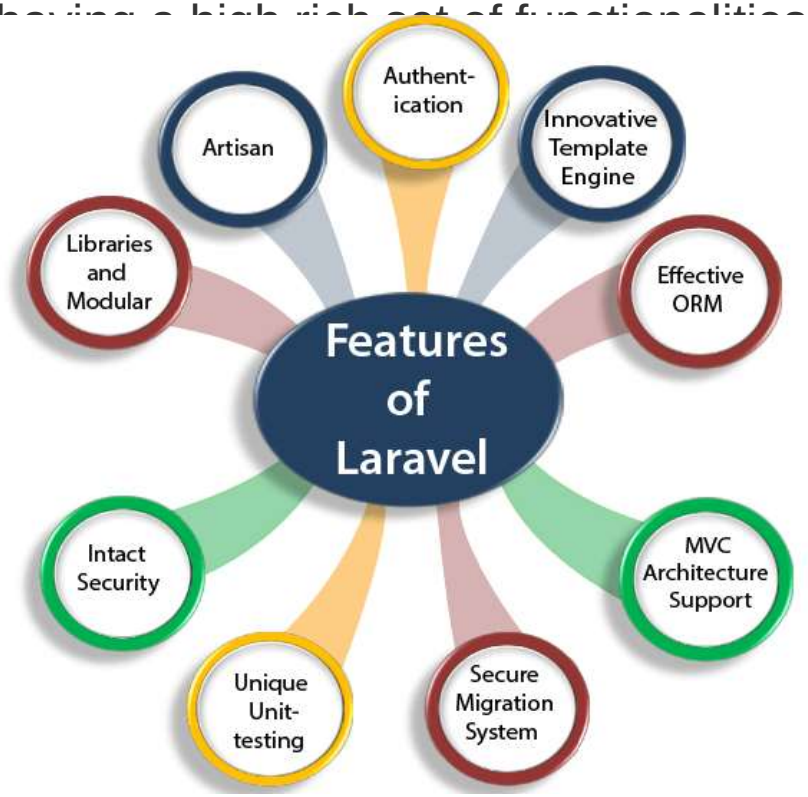
- Laravel is a PHP framework that uses the MVC architecture.

where,

- **Framework:** It is the collection of methods, classes, or files that the programmer uses, and they can also extend its functionality by using their code.
- **Architecture:** It is the specific design pattern that the framework follows. Laravel is following the MVC architecture.

Laravel Features

- We know that PHP is the oldest programming language used by the programmers, and more than 20 million websites are developed by using PHP. PHP is a very suitable programming language as it satisfies the business requirements whether the business is big or small. Laravel is one of the most popular frameworks having a high rich set of functionalities.



Continue..

- Authentication
- User authentication is a common feature in web applications. Laravel eases designing authentication as it includes features such as **register, forgot password** and **send password reminders**.
- Innovative Template Engine
- Laravel provides an innovative template engine which allows the developers to create the dynamic website. The available widgets in Laravel can be used to create solid structures for an application.
- Effective ORM (Object Relational Mapper)
- Laravel contains an inbuilt ORM with easy PHP Active Record implementation. An effective ORM allows the developers to query the database tables by using the simple PHP syntax without writing any SQL code. It provides easy integration between the developers and database tables by giving each of the tables with their corresponding models.

Continue..

MVC Architecture Support

- Laravel supports MVC architecture. It provides faster development process as in MVC; one programmer can work on the view while other is working on the controller to create the business logic for the web application. It provides multiple views for a model, and code duplication is also avoided as it separates the business logic from the presentation logic.

Secure Migration System

- **Laravel framework** can expand the database without allowing the developers to put much effort every time to make changes, and the migration process of Laravel is very secure and full-proof. In the whole process, **php code** is used rather than **SQL code**.

Unique Unit-testing

- Laravel provides a unique unit-testing. Laravel framework can run several test cases to check whether the changes harm the web app or not. In Laravel, developers can also write the test cases in their own code.

Continue..

Intact Security

- Application security is one of the most important factors in web application development. While developing an application, a programmer needs to take effective ways to secure the application. Laravel has an inbuilt web application security, i.e., it itself takes care of the security of an application. It uses "Bcrypt Hashing Algorithm" to generate the salted password means that the password is saved as an encrypted password in a database, not in the form of a plain text.

Libraries and Modular

- Laravel is very popular as some Object-oriented libraries, and pre-installed libraries are added in this framework, these pre-installed libraries are not added in other **php frameworks**. One of the most popular libraries is an **authentication library** that contains some useful features such as password reset, monitoring active users, Bcrypt hashing, and CSRF (**Cross-Site Request Forgery**) protection. This framework is divided into several modules that follow the php principles allowing the developers to build responsive and modular apps

Continue..

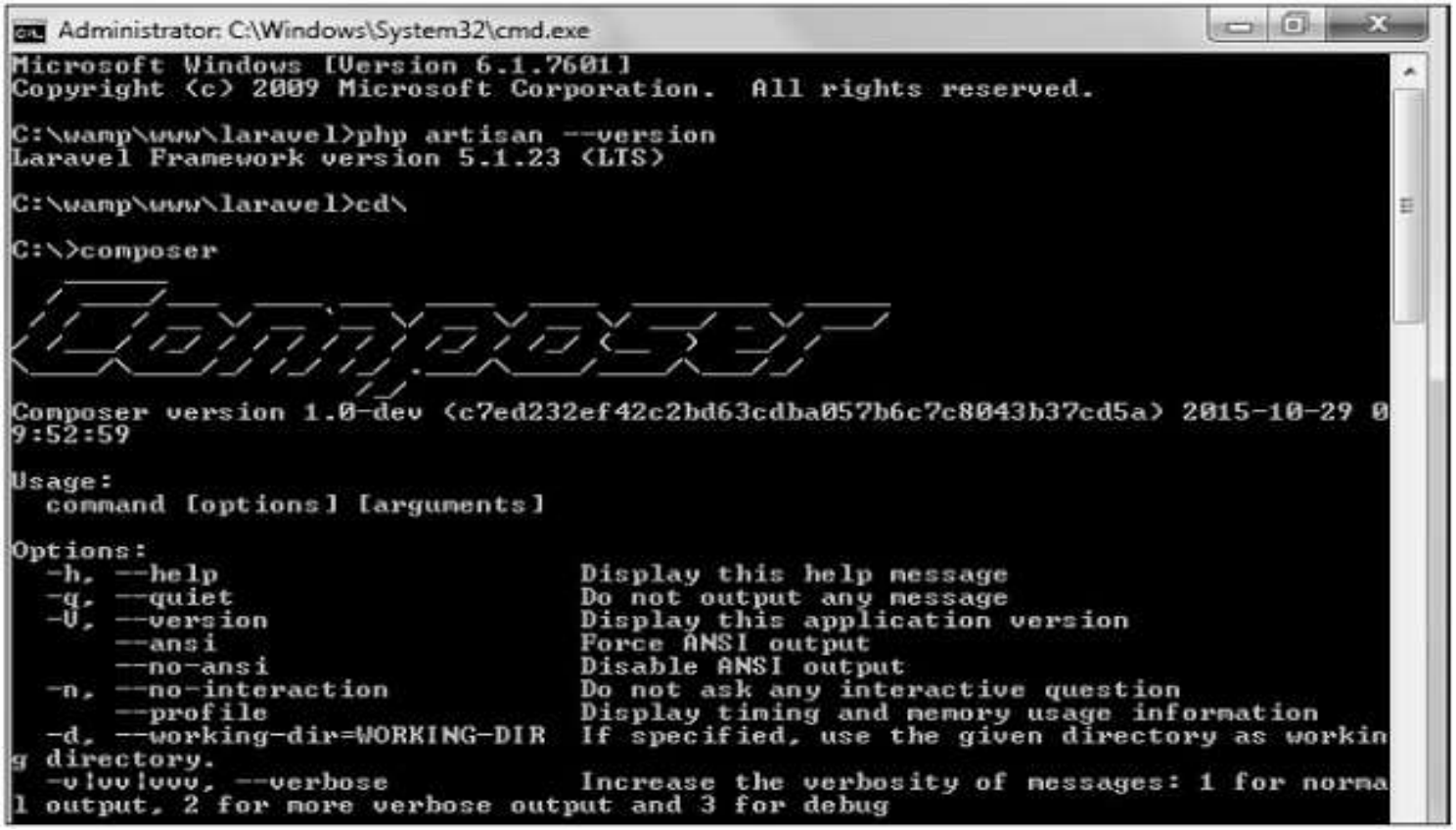
- Artisan
- Laravel framework provides a built-in tool for a command-line known as **Artisan** that performs the repetitive programming tasks that do not allow the php developers to perform manually. These artisans can also be used to create the skeleton code, database structure, and their migration, so it makes it easy to manage the database of the system. It also generates the MVC files through the command line. Artisan also allows the developers to create their own commands.

Laravel installation

- For managing dependencies, Laravel uses **composer**. Make sure you have a Composer installed on your system before you install Laravel. In this chapter, you will see the installation process of Laravel.
- You will have to follow the steps given below for installing Laravel onto your system −
- **Step 1** − Visit the following URL and download composer to install it on your system.
- https://getcomposer.org/download/
- **Step 2** − After the Composer is installed, check the installation by typing the Composer command in the command prompt as shown in the following screenshot.

Laravel installation

-



Laravel installation

- Step 3 – Create a new directory anywhere in your system for your new Laravel project. After that, move to path where you have created the new directory and type the following command there to install Laravel.

- composer create-project laravel/laravel –-prefer-dist
- Now, we will focus on installation of version 5.7. In Laravel version 5.7, you can install the complete framework by typing the following command –

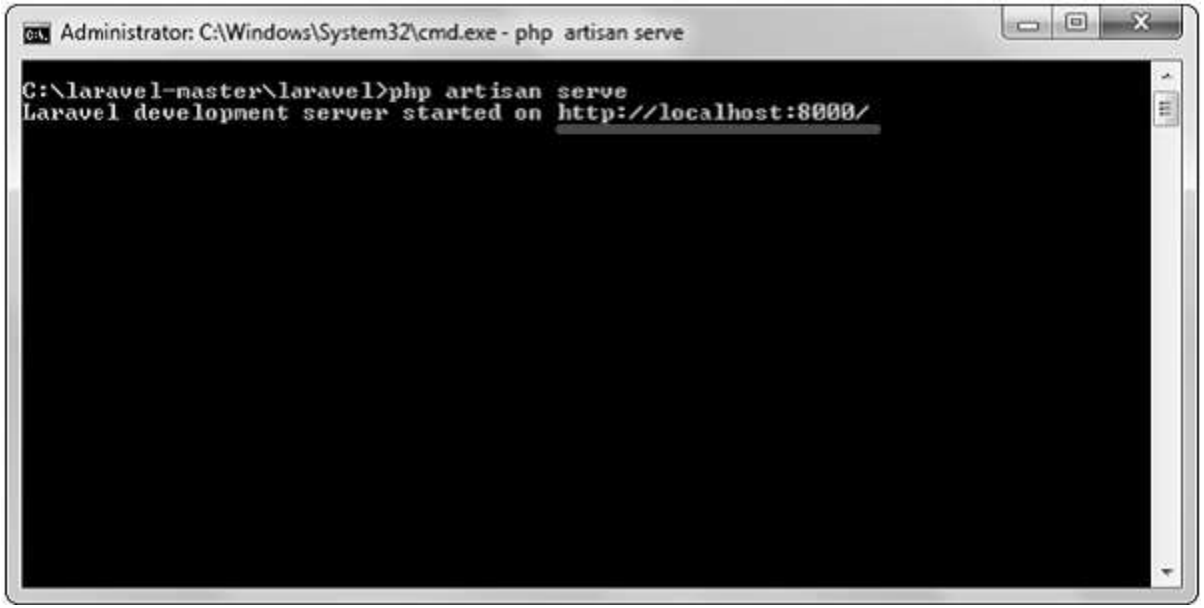
- composer create-project laravel/laravel test dev-develop
- The output of the command is as shown below –

Laravel installation

```
➜  code composer create-project laravel/laravel test dev-develop
Installing laravel/laravel (dev-develop d6acad21cb2288713d9c09a31f9b4ab86f116039)
  - Installing laravel/laravel (dev-develop develop): Cloning develop from cache
Created project in test
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 71 installs, 0 updates, 0 removals
  - Installing vlucas/phpdotenv (v2.5.1): Loading from cache
  - Installing symfony/css-selector (v4.1.3): Loading from cache
  - Installing tijsverkoyen/css-to-inline-styles (2.2.1): Loading from cache
  - Installing symfony/polyfill-php72 (v1.9.0): Loading from cache
  - Installing symfony/polyfill-mbstring (v1.9.0): Loading from cache
  - Installing symfony/var-dumper (v4.1.3): Loading from cache
  - Installing symfony/routing (v4.1.3): Loading from cache
  - Installing symfony/process (v4.1.3): Loading from cache
  - Installing symfony/polyfill-ctype (v1.9.0): Loading from cache
  - Installing symfony/http-foundation (v4.1.3): Loading from cache
  - Installing symfony/event-dispatcher (v4.1.3): Loading from cache
  - Installing psr/log (1.0.2): Loading from cache
  - Installing symfony/debug (v4.1.3): Loading from cache
  - Installing symfony/http-kernel (v4.1.3): Loading from cache
  - Installing paragonie/random_compat (v9.99.99): Loading from cache
```

Laravel installation

- The Laravel framework can be directly installed with develop branch which includes the latest framework.
- Step 4 – The above command will install Laravel in the current directory. Start the Laravel service by executing the following command.
- php artisan serve
- Step 5 – After executing the above command, you will see a screen as shown below –



Laravel installation

- **Step 6** – Copy the URL underlined in gray in the above screenshot and open that URL in the browser. If you see the following screen, it implies Laravel has been installed successfully.

Topic objective

- **Application Structure of Laravel**

In this topic student will understand the structure of Laravel, directory of Laravel and bootstrap directory.

Application Structure of Laravel

- The application structure in Laravel is basically the structure of folders, sub-folders and files included in a project. Once we create a project in Laravel, we get an overview of the application structure as shown in the image here.
- The snapshot shown here refers to the root folder of Laravel namely **laravel-project**. It includes various sub-folders and files. The analysis of folders and files, along with their functional aspects is given below −

Root Directory

- Application Structure

Continue..

- App
- It is the application folder and includes the entire source code of the project. It contains events, exceptions and middleware declaration. The app folder comprises various sub folders as explained below –
- Console
- Console includes the artisan commands necessary for Laravel. It includes a directory named **Commands**, where all the commands are declared with the appropriate signature. The file **Kernal.php** calls the commands declared in **Inspire.php**.

Continue..

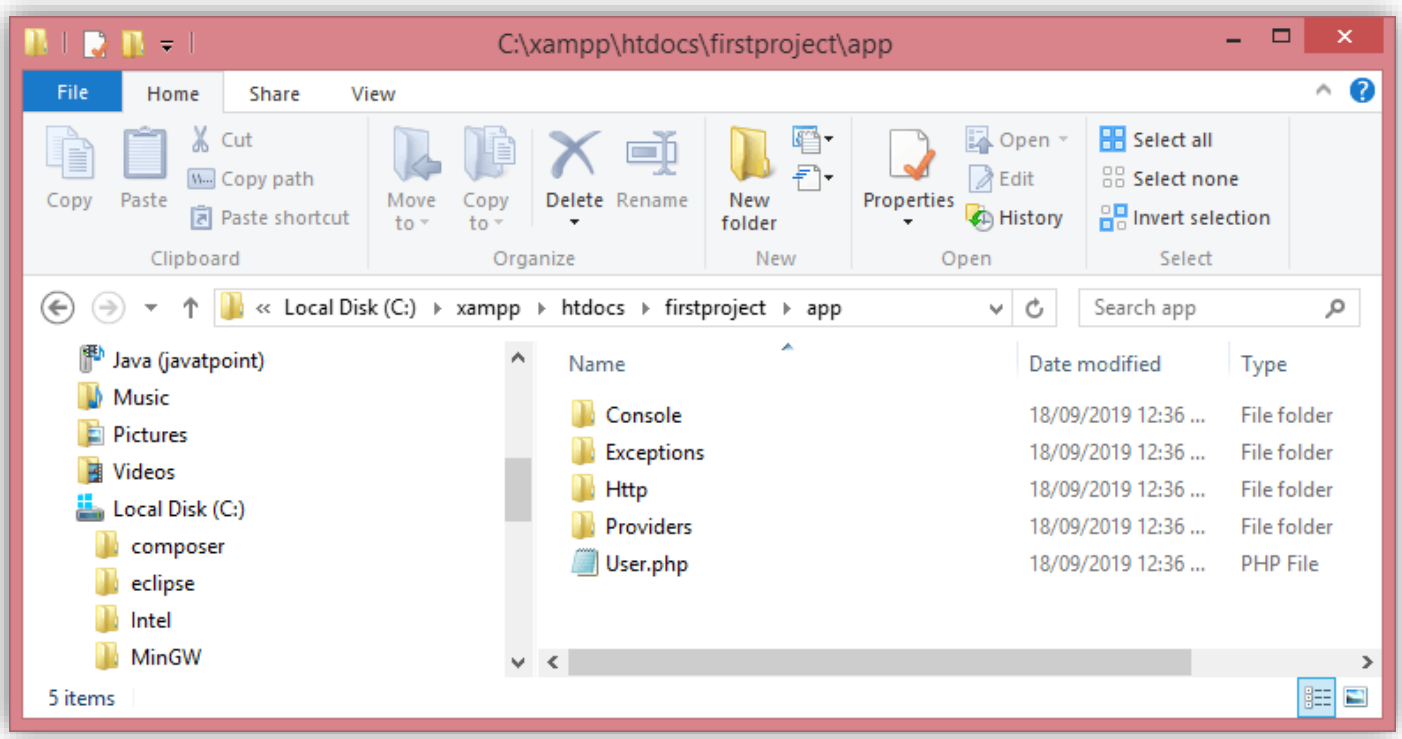
- If we need to call a specific command in Laravel, then we should make appropriate changes in this directory.

- Events
- This folder includes all the events for the project.

- Events
- Events are used to trigger activities, raise errors or necessary validations and provide greater flexibility. Laravel keeps all the events under one directory. The default file included is event.php where all the basic events are declared.

Continue..

- Laravel's app directory
- The app folder is one of the major folders in Laravel as most of the code is written in the app folder. The App folder contains the following sub-folders:
- Console
- Exceptions
- Http
- Providers



Continue..

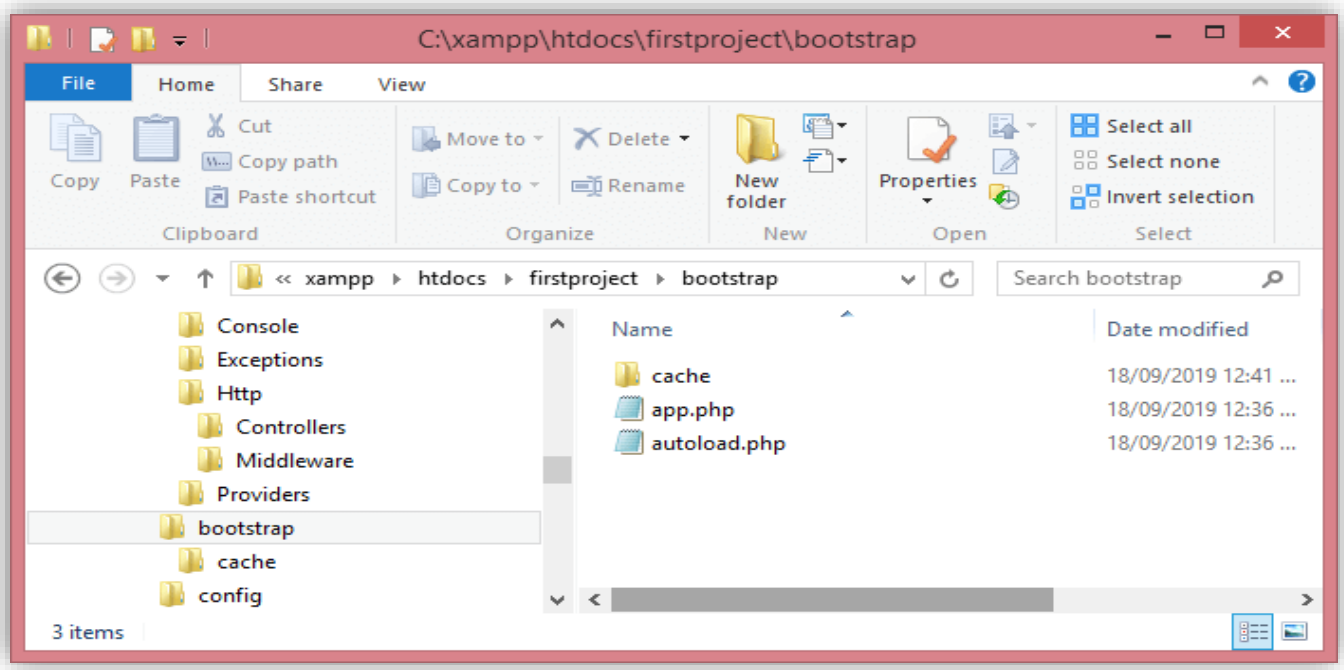
- Console
- Console folder contains the artisan commands required for Laravel. It contains the commands which are declared with the appropriate signature.
- Exceptions
- Exceptions folder contains the various exception handlers. It handles the exceptions thrown by the Laravel project. The Exceptions directory contains the methods that handle the exceptions.
- The Exceptions directory contains the file handle.php that handles all the exceptions.
- Http
- The http folder is a sub-folder of the app folder. It has sub-folders such as controllers, middleware, and requests. Laravel follows the MVC architecture, so http includes controllers, views, and requests.

Continue..

- Where,
- **Middleware:** It is a sub-folder of the http directory. It provides a filter mechanism and communication between request and response.
- **Requests:** It is a sub-folder of http which includes all the requests of an application.
- Providers
- The Providers directory is used to contain all the service providers that are required to register events for core servers and provides configuration for Laravel application.

Continue..

- Laravel's bootstrap directory
- The bootstrap directory holds the files that are required to bootstrap the Laravel application and to configure auto-loading. The bootstrap folder contains a sub-folder cache used for caching a web application. It also contains the file **app.php** that initializes the scripts required for bootstrap.

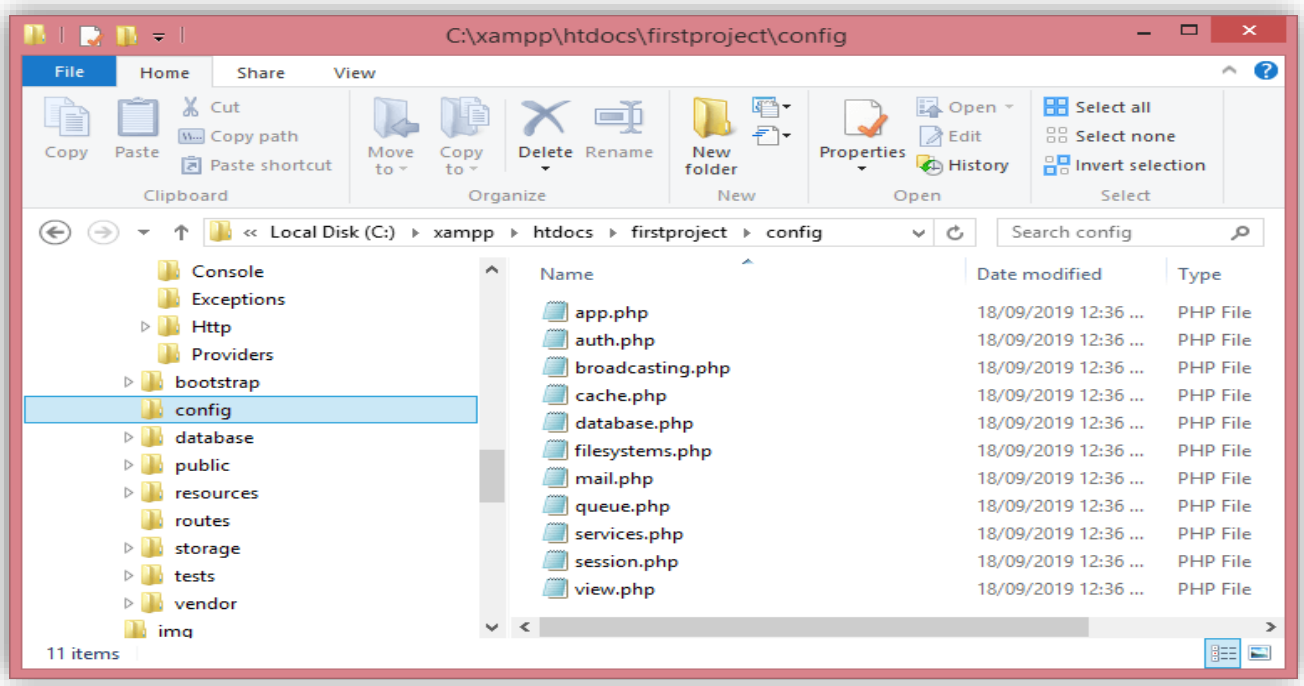


Continue..

- The above screen shows the structure of the bootstrap directory. It contains one folder, i.e., **cache** and two files, **app.php** and **autoload.php**.

Continue..

- Laravel's config directory
- The config's directory contains the various configuration files required for the Laravel application. Various files are available inside the **config's** directory shown in the below screenshot, and each file performs their functionalities as per their names.



Topic Objective

Topic :Basic Configuration

In this topic student will gain about basic configuration with Laravel and also how environmental configuration works.

Basic Configuration

- All of the configuration files for the Laravel framework are stored in the config directory. Each option is documented, so feel free to look through the files and get familiar with the options available to you.
- These configuration files allow you to configure things like your database connection information, your mail server information, as well as various other core configuration values such as your application timezone and encryption key.
- Application Overview
- We can get a quick overview of your application's configuration, drivers, and environment via the about Artisan command:

php artisan about

- If you're only interested in a particular section of the application overview output, you may filter for that section using the --only option:

php artisan about --only=environment

Environmental Configuration

Environment Configuration

- Environment Variable Types
- Retrieving Environment Configuration
- Determining The Current Environment
- Encrypting Environment Files

- Environment Configuration
- It is often helpful to have different configuration values based on the environment where the application is running. For example, you may wish to use a different cache driver locally than you do on your production server.
- To make this a effective, Laravel utilizes the DotEnv PHP library. In a fresh Laravel installation, the root directory of your application will contain a .env.example file that defines many common environment variables. During the Laravel installation process, this file will automatically be copied to .env.

Continue..

- Environment Variable Types
- All variables in your .env files are typically parsed as strings, so some reserved values have been created to allow you to return a wider range of types from the env() function:

.env Value	env() Value
true	(bool) true
(true)	(bool) true
false	(bool) false
(false)	(bool) false
empty	(string) ''
(empty)	(string) ''
null	(null) null
(null)	(null) null

Continue..

- If you need to define an environment variable with a value that contains spaces, you may do so by enclosing the value in double quotes:

```
APP_NAME="My Application"
```

Continue..

- **Retrieving Environment Configuration**
- All of the variables listed in the .env file will be loaded into the $_ENV PHP super-global when your application receives a request. However, you may use the env function to retrieve values from these variables in your configuration files. In fact, if you review the Laravel configuration files, you will notice many of the options are already using this function:

```
'debug' => env('APP_DEBUG', false),
```

- The second value passed to the env function is the "default value". This value will be returned if no environment variable exists for the given key.

Continue..

- Determining The Current Environment
- The current application environment is determined via the APP_ENV variable from your .env file. You may access this value via the environment method on the App facade:

```
use Illuminate\Support\Facades\App;

$environment = App::environment();
```

- You may also pass arguments to the environment method to determine if the environment matches a given value. The method will return true if the environment matches any of the given values:

```
if (App::environment('local')) {
    // The environment is local
}

if (App::environment(['local', 'staging'])) {
    // The environment is either local OR staging...
}
```

Continue..

- Encrypting Environment Files
- Unencrypted environment files should never be stored in source control. However, Laravel allows you to encrypt your environment files so that they may be safely be added to source control with the rest of your application.

- Encryption
- To encrypt an environment file, you may use the env:encrypt command:

- php artisan env:encrypt

- Running the env:encrypt command will encrypt your .env file and place the encrypted contents in an .env.encrypted file. The decryption key is presented in the output of the command and should be stored in a secure password manager. If you would like to provide your own encryption key you may use the --key option when invoking the command:

Continue..

- php artisan env:encrypt --key=3UVsEgGVK36XN82KKeyLFMhvosbZN1aF

- The length of the key provided should match the key length required by the encryption cipher being used. By default, Laravel will use the AES-256-CBC cipher which requires a 32 character key. You are free to use any cipher supported by Laravel's encrypter by passing the --cipher option when invoking the command.

- If your application has multiple environment files, such as .env and .env.staging, you may specify the environment file that should be encrypted by providing the environment name via the --env option:

- php artisan env:encrypt --env=staging

Topic Objective

Topic :Routing

Routing in Laravel allows the students to route all your application requests to its appropriate controller.

Routing

Routing is one of the essential concepts in Laravel. The main functionality of the routes is to route all your application requests to the appropriate controller.

Default Route files

All Laravel routes are defined inside the route files located in the **routes** directory.
When we create a project, then a route directory is created inside the project.
The **route/web.php** directory contains the definition of route files for your web interface.
The routes in web.php are assigned with the web middleware group that provides the features like session state and CSRF protection.
The routes defined in **routes/api.php** are assigned with the **API** middleware group, and they are stateless.

We will start by defining the routes in **routes/web.api** file.

Continue..

```
<?php
Route::get('/', function ()
 {
return view ('welcome');
});
```

In the above case, Route is the class which defines the static method get(). The get() method contains the parameters '/' and function() closure. The '/' defines the root directory and function() defines the functionality of the get() method.

In the above route, the url is '/'; therefore, we entered the **localhost/laravelproject/public URL** in the web browser.

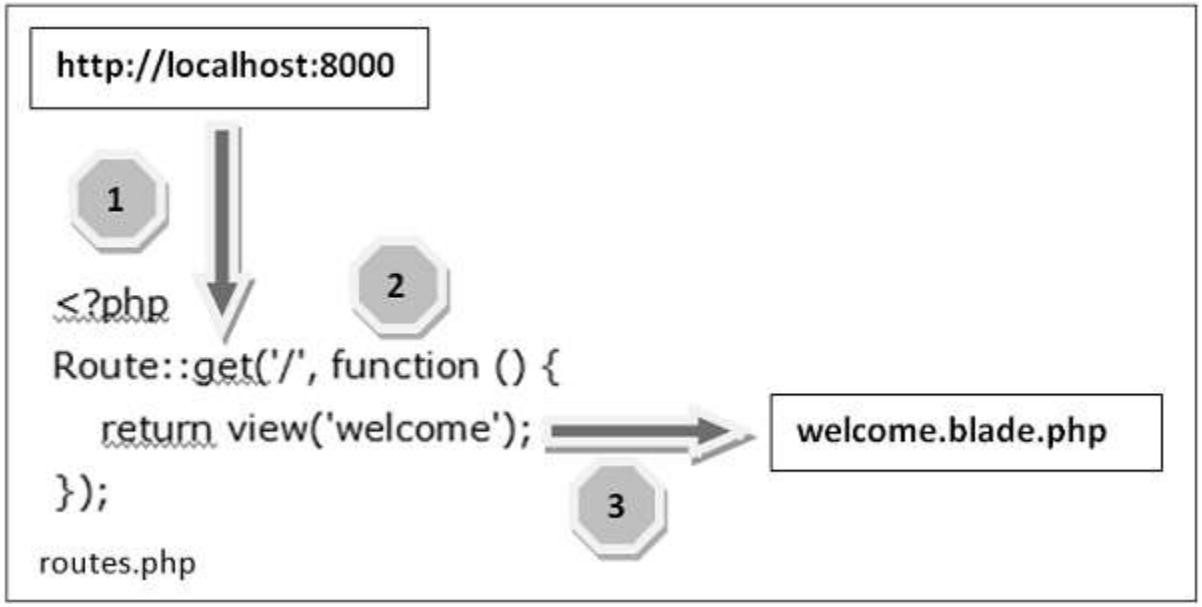
As the method returns the **view('welcome')**, so the above output shows the welcome view of the Laravel.

Continue..

```
<?php
Route::get('/example', function ()
 {
return "Hello World";
});
```

Continue..

- The routing mechanism is shown in the image given below –

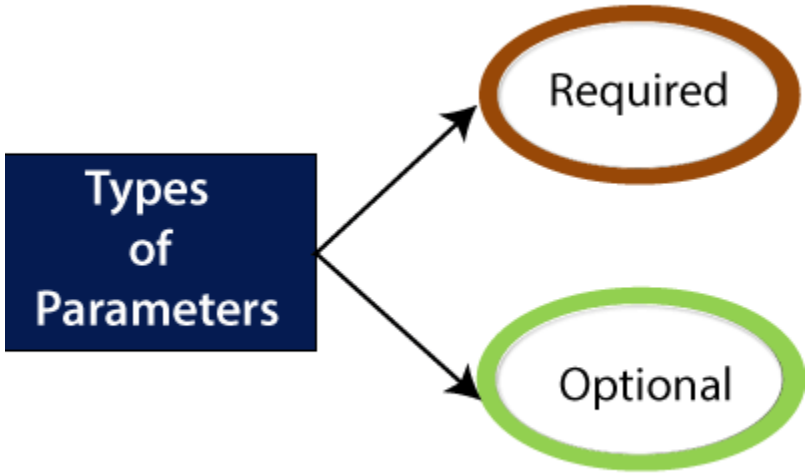


Continue..

- Let us now understand the steps involved in routing mechanism in detail –
- **Step 1** – Initially, we should execute the root URL of the application.
- **Step 2** – Now, the executed URL should match with the appropriate method in the **route.php** file. In the present case, it should match the method and the root (‘/’) URL. This will execute the related function.
- **Step 3** – The function calls the template file **resources/views/welcome.blade.php.** Next, the function calls the **view()** function with argument **‘welcome’** without using the **blade.php**.
- This will produce the HTML output as shown in the image below –

Routing Parameters

- **There are two types of parameters we can use:**
- **Required Parameters**
- **Optional Parameters**

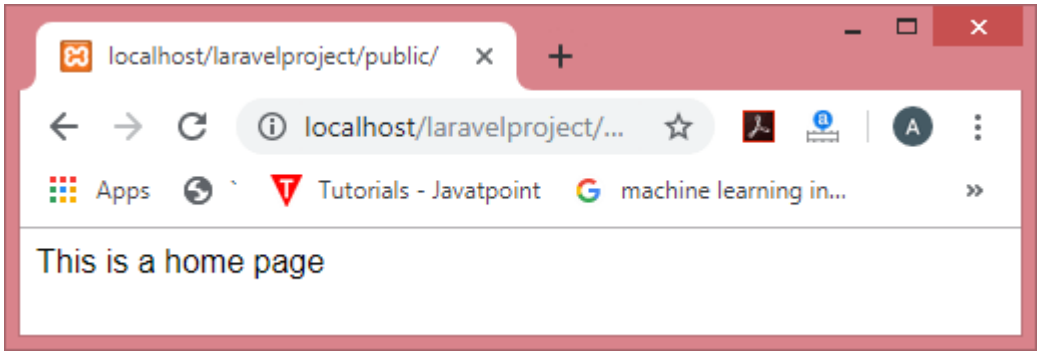


Continue..

- Required Parameters
- The required parameters are the parameters that we pass in the URL. Sometimes you want to capture some segments of the URI then this can be done by passing the parameters to the URL. For example, you want to capture the user id from the URL.

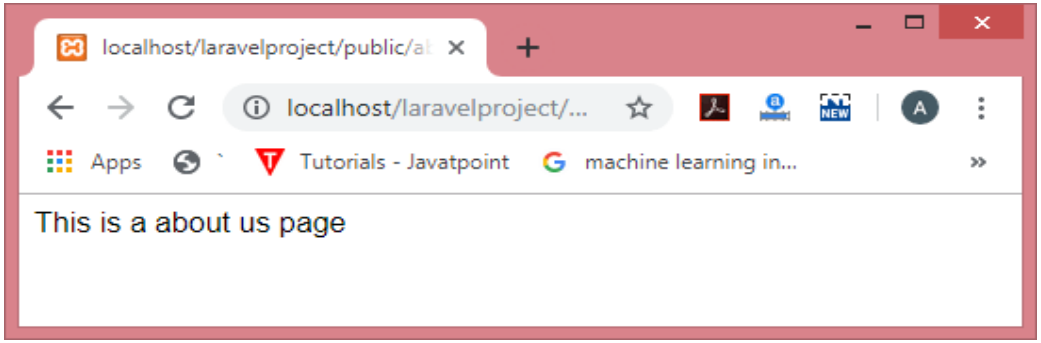
```
<?php
Route::get('/', function()
{
  return "This is a home page";
}
);
Route::get('/about', function()
{
  return "This is a about us page";
}
);
Route::get('/contact', function()
{
  return "This is a contact us page";
}
);
```

- **Output**
- **When we enter the URL "localhost/laravelproject/public/".**

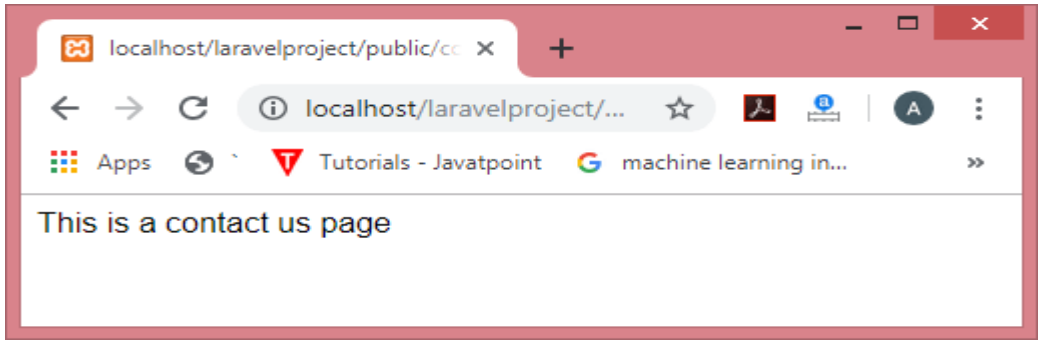


Continue..

- **When we enter the URL "localhost/laravelproject/public/about".**



- **When we enter the URL "localhost/laravelproject/public/contact".**



Optional Parameters

- Suppose you want to specify the route parameter occasionally, in order to achieve this, you can make the route parameter optional. To make the route parameter optional, you can place '?' operator after the parameter name. If you want to provide the optional parameter, and then make sure that you have also provided the default value to the variable.

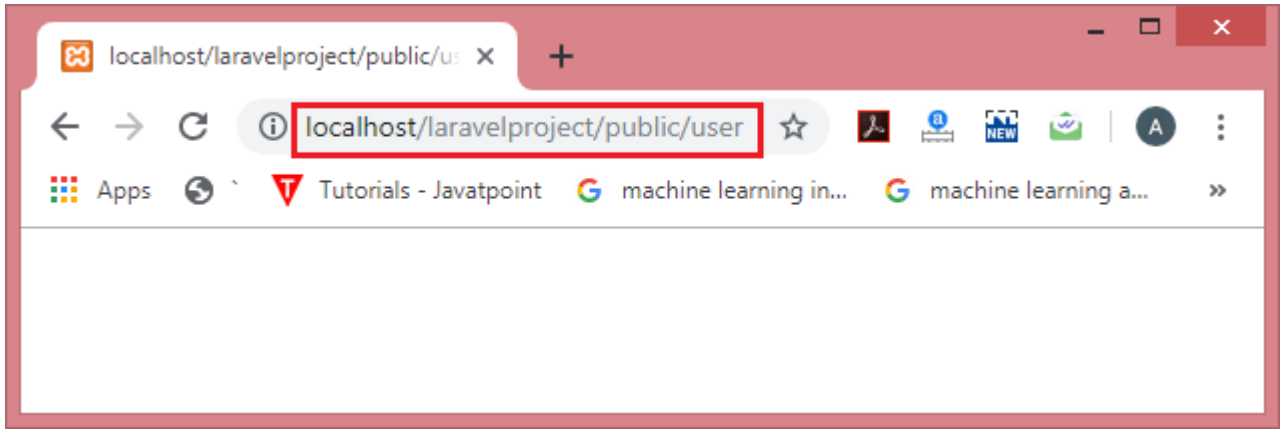
- **Let's understand through some examples.**

Example 1:

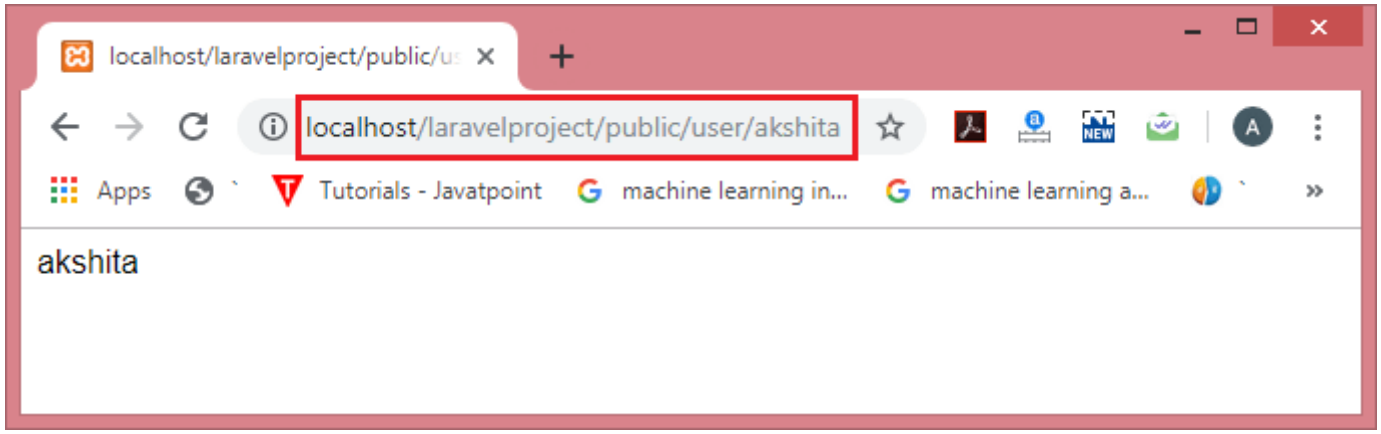
```
Route::get('user/{name?}', function ($name=null) {
    return $name;
});
```

Continue..

- When we do not pass any variable to the URL, then the output would be:



- When we pass '**akshita**' in the URL, then the output would be:

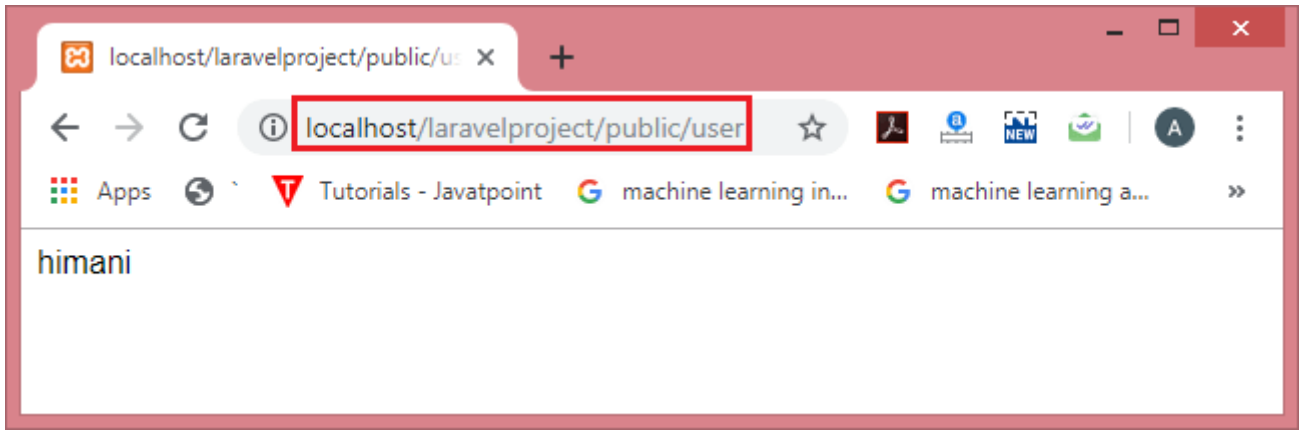


Continue..

- From the above outputs, we observe that the parameter we pass in the URL is optional. As we have provided the default value to the parameter as Null, so if we do not pass any parameter, it will return null. If we pass the parameter in the URL, then the value of the parameter would be displayed.

```
Route::get('user/{name?}', function ($name = 'himani') {
    return $name;
});
```

- In the above example, we have provided the default value as 'himani'.
- OUTPUT



Middleware

Topic : Middleware

In this topic student understand that how Middleware provide a convenient mechanism for inspecting and filtering HTTP requests entering your application. For example, Laravel includes a middleware that verifies the user of your application is authenticated.

Middleware

Middleware acts as a layer between the user and the request. It means that when the user requests the server then the request will pass through the middleware, and then the middleware verifies whether the request is authenticated or not. If the user's request is authenticated then the request is sent to the backend. If the user request is not authenticated, then the middleware will redirect the user to the login screen.

An additional middleware can be used to perform a variety of tasks except for authentication. For example, CORS (Cross Origin Resource Sharing) middleware is responsible for adding headers to all the responses.

Laravel framework includes several middleware such as authentication and CSRF protection, and all these are located in the app/Http/Middleware directory.
We can say that middleware is an http request filter where you can check the conditions.

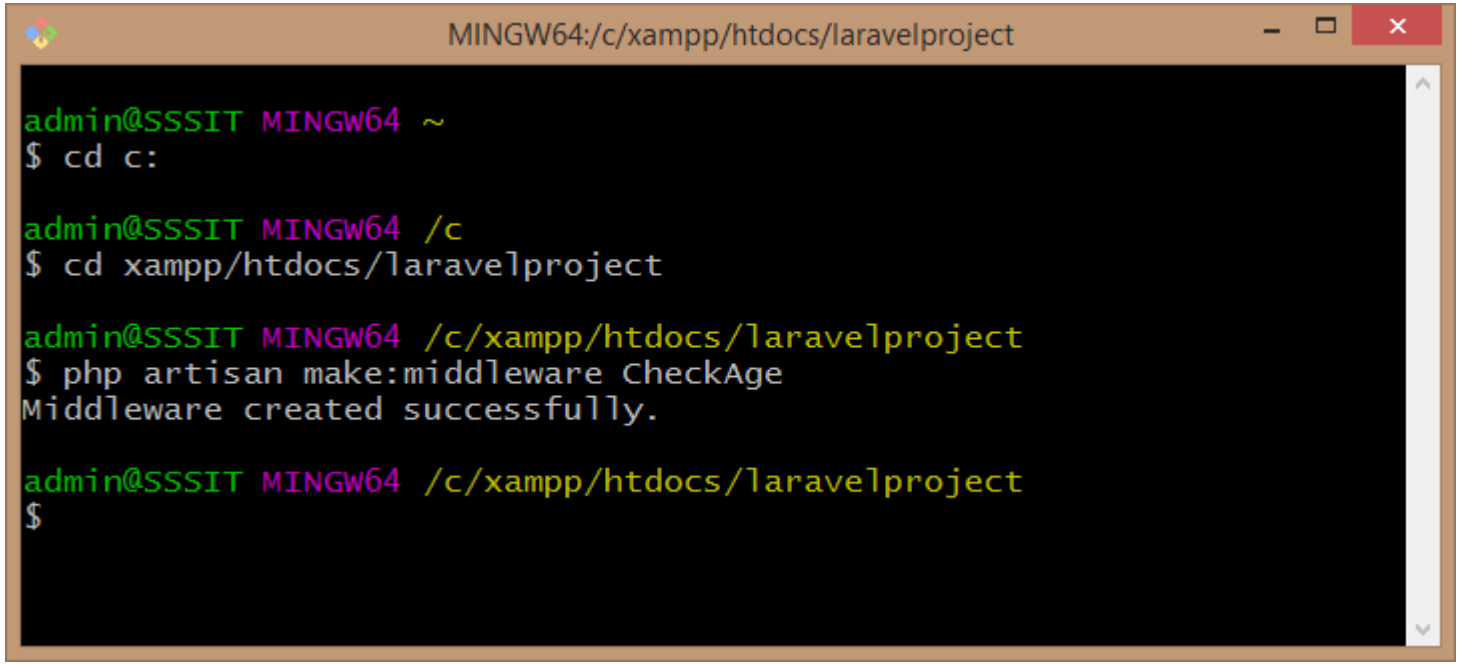
Middleware Parameters

In middleware, we are going to discuss the following topics:

- Make a middleware
- Apply middleware
- Check condition in middleware
- Route middleware

Creating a Middleware

Type the command **php artisan make:middleware 'name of the middleware'**.



Creating a Middleware

In the above screen, we type the command "**php artisan make:middleware CheckAge**" where **CheckAge** is the name of the middleware. The above window shows that the middleware has been created successfully with the name "**CheckAge**".

To see whether the CheckAge middleware is created or not, go to your project. Our project name is laravelproject, so the path for the middleware would be: **C:\xampp\htdocs\laravelproject\app\Http\Middleware**.

Apply a Middleware

Middleware can be either applied to all the URLs or some particular URLs.

Let's apply the middleware to all the URLs.

Step 1: Open the kernel.php file. If we want to apply the middleware to all the URLs, then add the path of the middleware in the array of middleware.

Step 2: Type the command **php artisan serve** in Git Bash Window.

Step 3: Open the **CheckAge.php** file, which you have created as a middleware.

Step 4: Now, enter the URL 'http://localhost/laravelproject/public/'.

Apply a Middleware

Let's apply the middleware to some specific routes.

Step 1: Open the kernel.php file. If we want to apply the middleware to some specific routes

Step 2: Open the **CheckAge.php** file, which you have created as a middleware.

Step 3: Add the middleware code in the **web.php** file.

```
Route::Get('/',function()
{
  return view('welcome');
})-> middleware('age');
Route::Get('user/profile',function()
{
  return "user profile";
});
```

Controllers in Laravel

Controllers are used to handle the request logic within the single class, and the controllers are defined in the "app/http/Controllers" directory.

Laravel framework follows the MVC (Model View Controller) architecture in which controllers act as moving the traffic back and forth between model and views.

The default file of controller is available in the **app/http/Controllers** directory.

In code, the namespace is used as it allows you to use the same function names and classes in the different parts of the same application. For example,

```
namespace App\Http\functions1;
namespace App\Http\functions2;
```

Cont…

Suppose we have to run the function having the name, i.e., RunQuery(). They are available in different directories functions1 and functions2, so we can say that namespace avoids the collision between the same function names.

'use' is used to import the class to the current file.

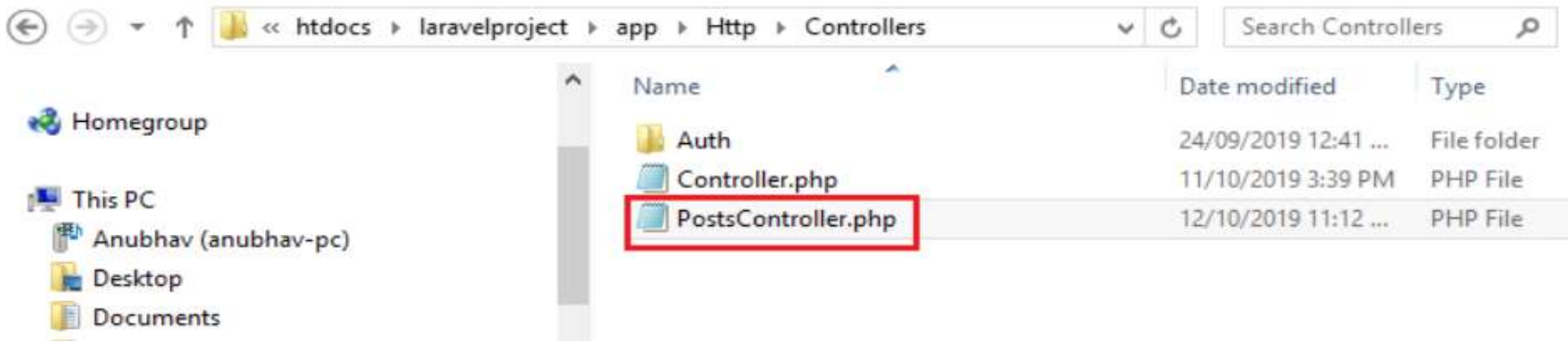
how to create the controller through Git Bash Window.

Step 1: Open the Git Bash Window and type the command "**php artisan make:Controller PostsController**" in Git Bash Window to create the Controller.

Step 2: Now move to your project and see whether the PostsController file has been created or not. The path of the file is:

C:\xampp\htdocs\laravelproject\app\Http\Controllers

Cont…



The above screenshot shows that the PostsController file is created.

You can check the default code of PostsController.php.

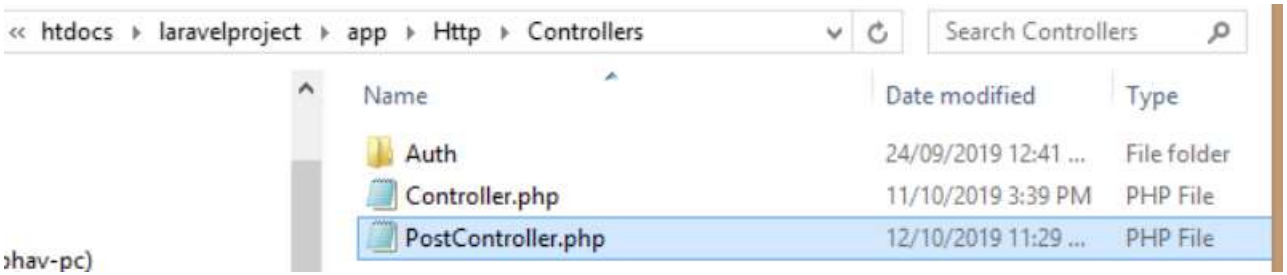
The code contains the class that extends the Controller class, but this class does not contain the functions such as create, update, or delete. Now we will see how to create the controller which contains some default functions.

Cont…

To create the Controller, we will first delete the PostsController.php from the project, which we have created in the previous step.

php artisan make:controller -resource PostController
this command is used to create the controller.

C:\xampp\htdocs\laravelproject\app\Http\Controllers



The above screen shows that PostController file is created successfully

Check default code of the PostController.php file.

Cont…

The code contains the functions which are used to perform the various operations on the resources such as:

create(): It is used to create a new resource.

store(): It is used to store the specified resource.

update(): It is used to update the specified resource in the storage

destroy(): It is used to remove the specified resources from the storage

RESTful Resource Controllers

Laravel resource controllers provide the CRUD routes to the controller in a single line of code. A resource controller is used to create a controller that handles all the http requests stored by your application.
The resource() is a static function like get() method that gives access to multiple routes that we can use in a controller.

Syntax of resource() method:
Route::resource('posts','PostController');

In the above syntax, 'posts' contains all the routes, and 'PostController' is the name of the controller. In this case, we do not need to specify the method name such as @index as we did in get() method because create(), store(), destroy() methods are already available in the PostController class.

RESTful Resource Controllers

Step 1: Create the controller by using the command given below:

```
php  artisan make:controller PostController
```

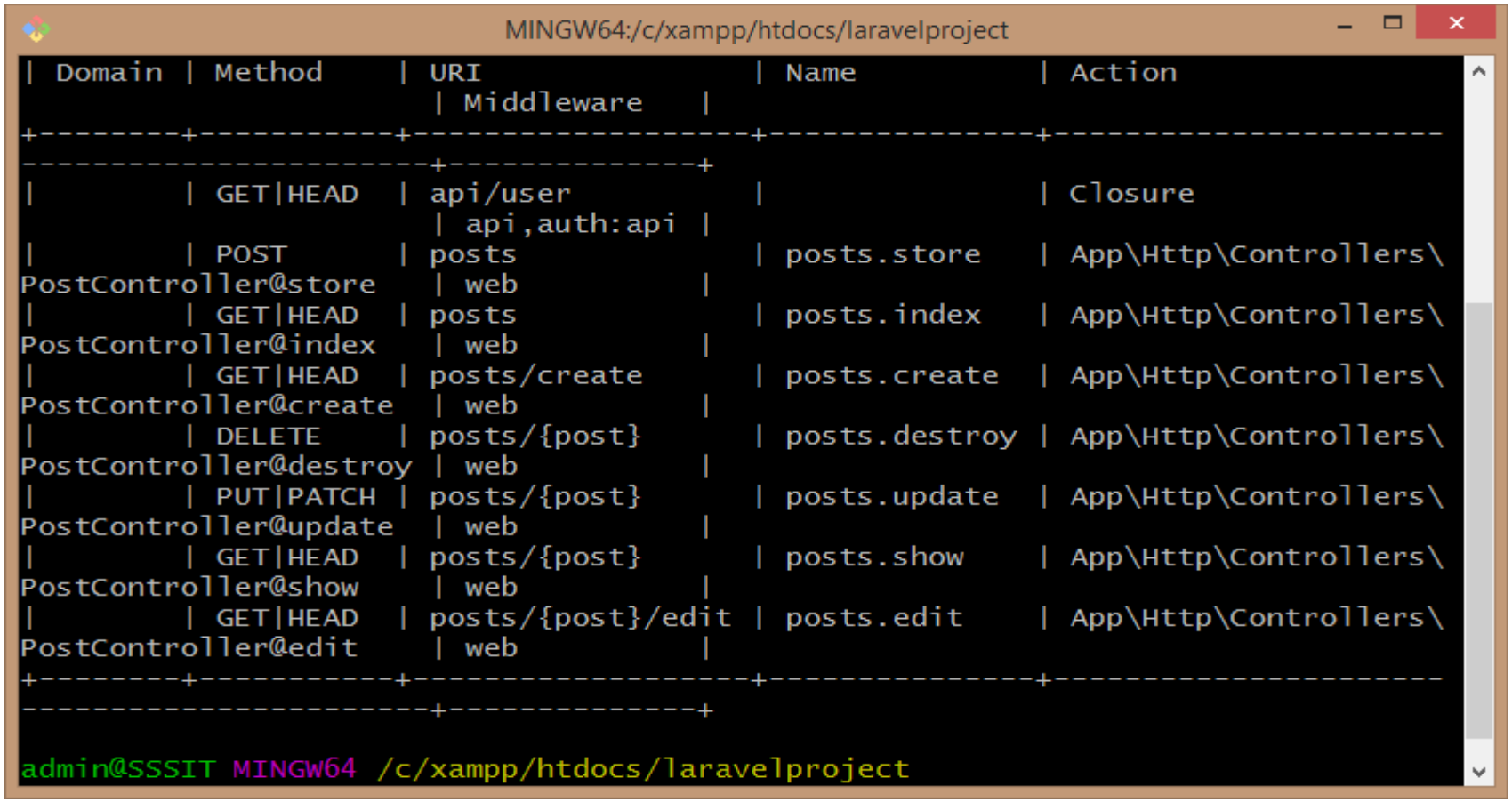
The above command will create the Controller at the **app/Http/Controllers/PostController.php** directory. The **PostController** class contains the methods for each resource operations.

Step 2: Now, we need to register the resourceful route to the Controller, and which can be done as follows:

```
Route::resource('posts','PostController');
```

RESTful Resource Controllers

Open the Git Bash Window, and enter the command **php artisan route:list**. This command produces the following output:



RESTful Resource Controllers

The post parameter in the resource() method produces the names or resources shown in the above output, and its corresponding methods. In the above output, the posts.destroy is sending a parameter to the Delete method, which is very special in Laravel.

Let's understand the concept of resources through an example.

Accessing the show() method of PostController class

Suppose we want to call the **show()** method of PostController.php file.
To do so, add the code in show() method. I added the following code in show() method:

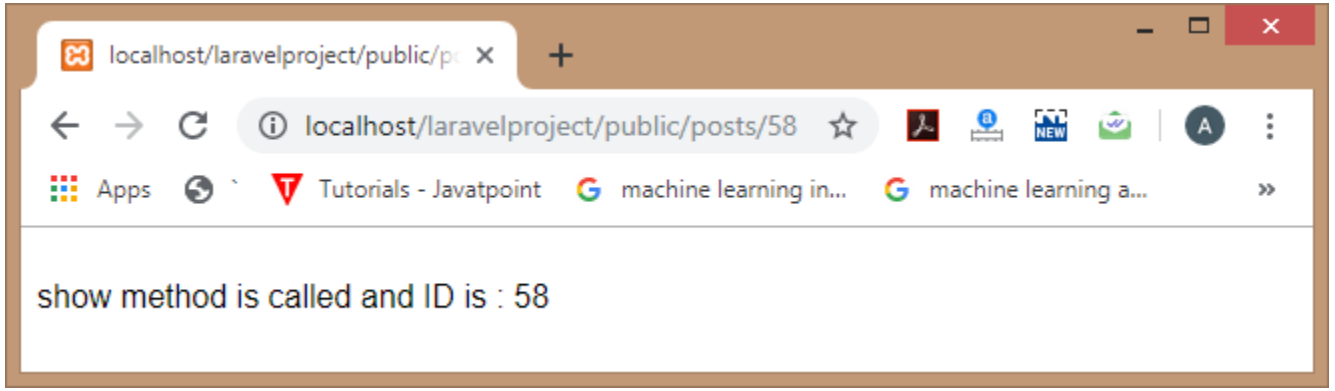
RESTful Resource Controllers

```
public function show($id)
{
return "show method is called and ID is : ".
$id
}
```

As we know that URI of the **posts.show** is posts/{posts}, which means that we need to enter the parameter as well to access the show() method of the **PostController** class.

Suppose I entered the URL as '**localhost/laravelproject/public/posts/58**', then the output would be:

RESTful Resource Controllers



Accessing the create() method of PostController class

Step 1: First, we need to add the code in create() method.

RESTful Resource Controllers

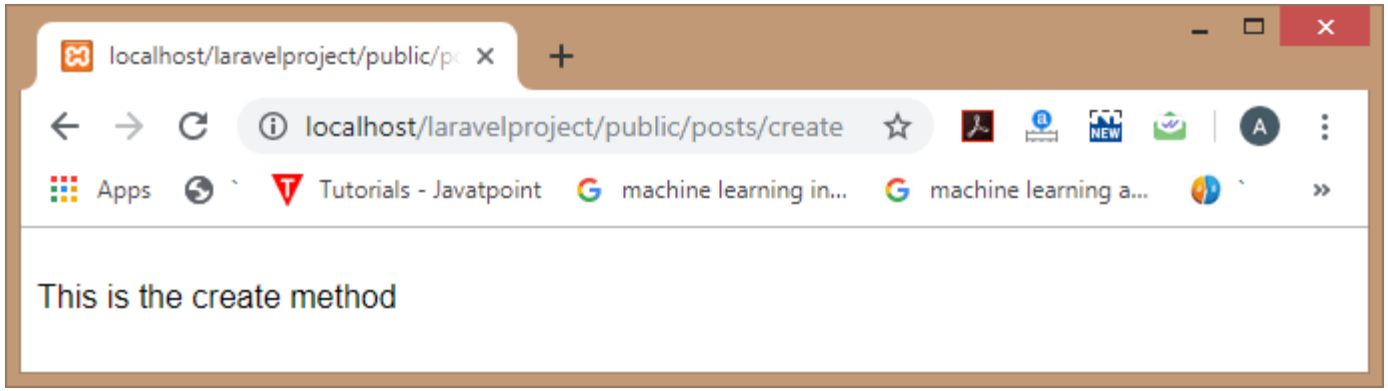
```
public function create()
{
    return "This is the create method";
}
```

As we know that the URI of the **posts.create** is posts/create, so the URL to access the create() method would be

'**localhost/laravelproject/public/posts/create**'.

RESTful Resource Controllers

Step 2: Enter the URL '**localhost/laravelproject/public/posts/create**' to the browser, then the output would be:



RESTful Resource Controllers

Registering routes for multiple controllers

We can register the routes for multiple controllers by passing an array to the **resources()** method. Suppose I want to register the routes for two controllers, such as **PostController** and **StudentController**. Following are the steps to achieve this:

Step 1: First, you need to create the **PostController** and **StudentController** by using the following commands:

```
Php artisan make:controller PostController;
  // to create the PostController.

Php artisan make:controller StudentControl
ler;  // to create the StudentController.
```

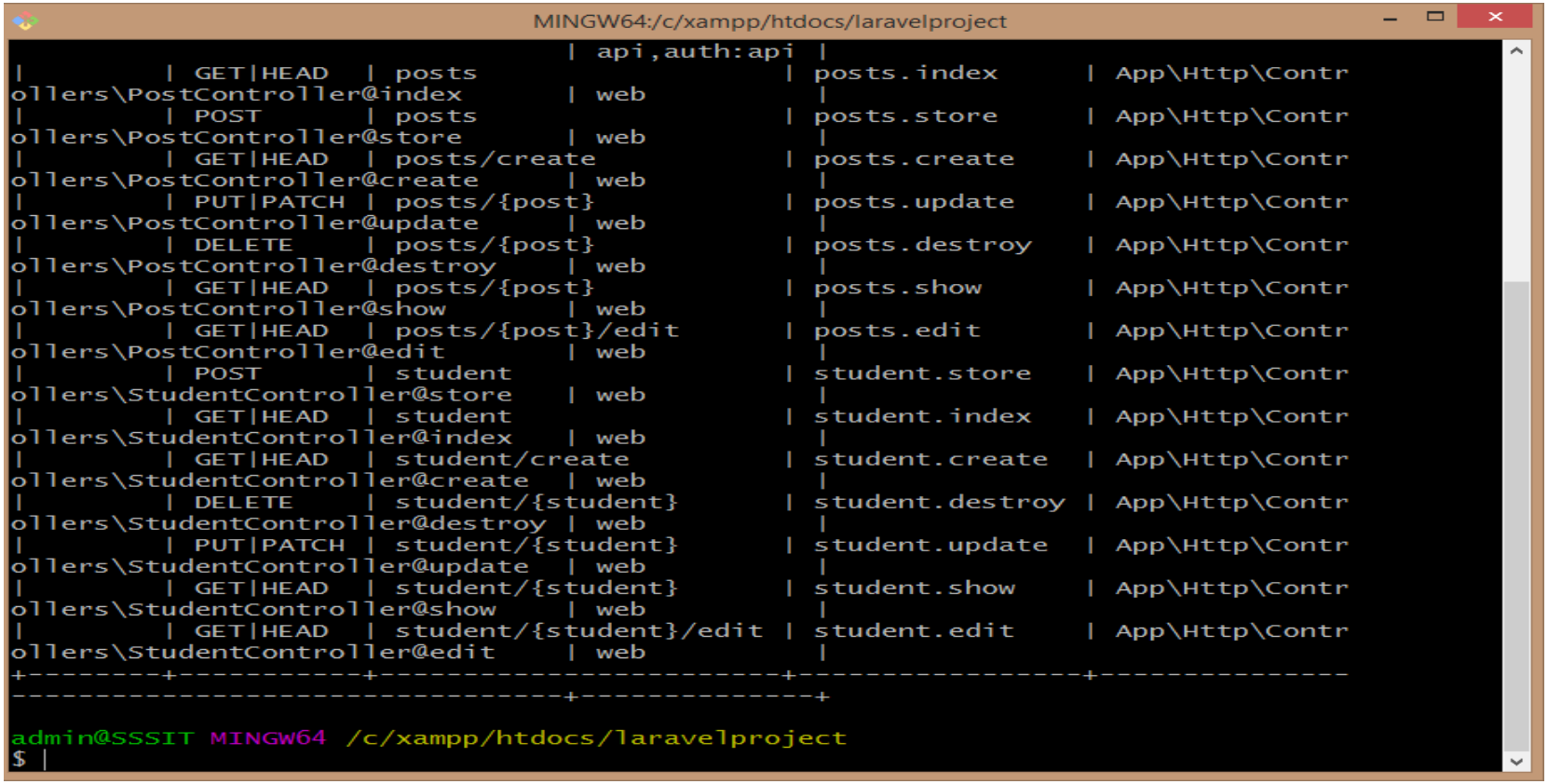
RESTful Resource Controllers

Step 2: Add the code given below in **web.php** file to register routes:

```
route::resources(
['posts'=>'PostController',
'student'=>'StudentController']
);
```

Step 3: Enter the command **php artisan route:list** on Git Bash Window.

RESTful Resource Controllers



The above screen shows that routes of both the PostController and StudentController are registered.

Dependency Injection & Controllers

Constructor Injection:

The Laravel service container is used to resolve all Laravel controllers. As a result, you are able to type-hint any dependencies your controller may need in its constructor. The declared dependencies will automatically be resolved and injected into the controller instance:

Constructor Injection in Laravel

Constructor injection in Laravel is a form of dependency injection where dependencies are injected into a class through its constructor. This approach allows for the inversion of control, where the responsibility of creating and managing dependencies is shifted to an external entity, typically Laravel's service container.

1. Define Dependencies in the Constructor: In your class, you specify the dependencies it requires in its constructor. These dependencies are typically other classes or interfaces that the class relies on to perform its tasks

In this example, the UserService class depends on the UserRepository class. The UserRepository instance is injected into UserService through its constructor.

Cont…

```
use App\Repositories\UserRepository;

class UserService
{
   protected $userRepository;

   public function
__construct(UserRepository
$userRepository)
   {
      $this->userRepository =
$userRepository;
   }

   // Other methods using $this-
>userRepository...
}
```

Cont…

2. Dependency Resolution by Laravel's Service Container: When an instance of the class is requested (either directly or as a dependency of another class), Laravel's service container resolves the dependencies specified in the constructor and automatically injects them.

3. Binding Dependencies: Before constructor injection can work, dependencies must be registered with Laravel's service container. Laravel provides various ways to bind dependencies, such as using service providers, the AppServiceProvider, or directly in the container.

For example, you might bind UserRepository in the AppServiceProvider:

This binds UserRepository to a concrete implementation, ensuring that Laravel knows how to resolve it when injected as a dependency.

Cont…

```
use App\Repositories\UserRepository;

class AppServiceProvider extends
ServiceProvider
{
   public function register()
   {
      $this->app-
>bind(UserRepository::class, function
($app) {
          return new UserRepository();
      });
   }
}
```

Benefits of Constructor Injection

Testability: Constructor injection makes classes more testable by allowing dependencies to be easily replaced with mock objects during unit testing. This facilitates isolated testing of classes without needing to interact with their actual dependencies.

Decoupling: Constructor injection promotes loose coupling between classes, as classes depend on abstractions (interfaces or abstract classes) rather than concrete implementations. This makes code more maintainable and flexible.

Readability: Explicitly declaring dependencies in the constructor improves the readability of code by clearly documenting the class's requirements and dependencies.

Method Injection in Laravel

Method injection in Laravel is a technique used for injecting dependencies into controller methods or class methods automatically by the Laravel service container.
This feature allows you to type-hint dependencies in your controller methods, and Laravel will automatically resolve those dependencies and pass them to the method when it is called.

Suppose you have a UserController with a method show that requires an instance of UserService:

In this example, the show method of the UserController class receives an instance of UserService through method injection. Laravel's service container will automatically resolve and inject the UserService instance when the show method is called.

Cont…

```
namespace App\Http\Controllers;

use App\Services\UserService;

class UserController extends Controller
{
   public function show(UserService
$userService, $userId)
   {
      $user = $userService-
>getUserById($userId);
      return view('user.show', ['user' =>
$user]);
   }
}
```

Cont…

you'll need to ensure that UserService is registered in Laravel's service container, typically in the AppServiceProvider or a service provider dedicated to your services.

bind UserService in the AppServiceProvider:

```
namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use App\Services\UserService;

class AppServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(UserService::class, function ($app) {
            return new UserService(/* dependencies may be passed here */);
        });
    }
}
```

Laravel Sail

- Laravel Sail is a light-weight command-line interface for interacting with Laravel's default Docker development environment. Sail provides a great starting point for building a Laravel application using PHP, MySQL, and Redis without requiring prior Docker experience.

- At its heart, Sail is the docker-compose.yml file and the sail script that is stored at the root of your project. The sail script provides a CLI with convenient methods for interacting with the Docker containers defined by the docker-compose.yml file.

- Laravel Sail is supported on macOS, Linux, and Windows (via WSL2).

Cont…

Laravel Sail is automatically installed with all new Laravel applications so you may start using it immediately. To learn how to create a new Laravel application, please consult Laravel's installation documentation for your operating system.

During installation, you will be asked to choose which Sail supported services your application will be interacting with.

Installing Sail Into Existing Applications

If you are interested in using Sail with an existing Laravel application, you may simply install Sail using the Composer package manager. Of course, these steps assume that your existing local development environment allows you to install Composer dependencies:

```
composer require laravel/sail --dev
```

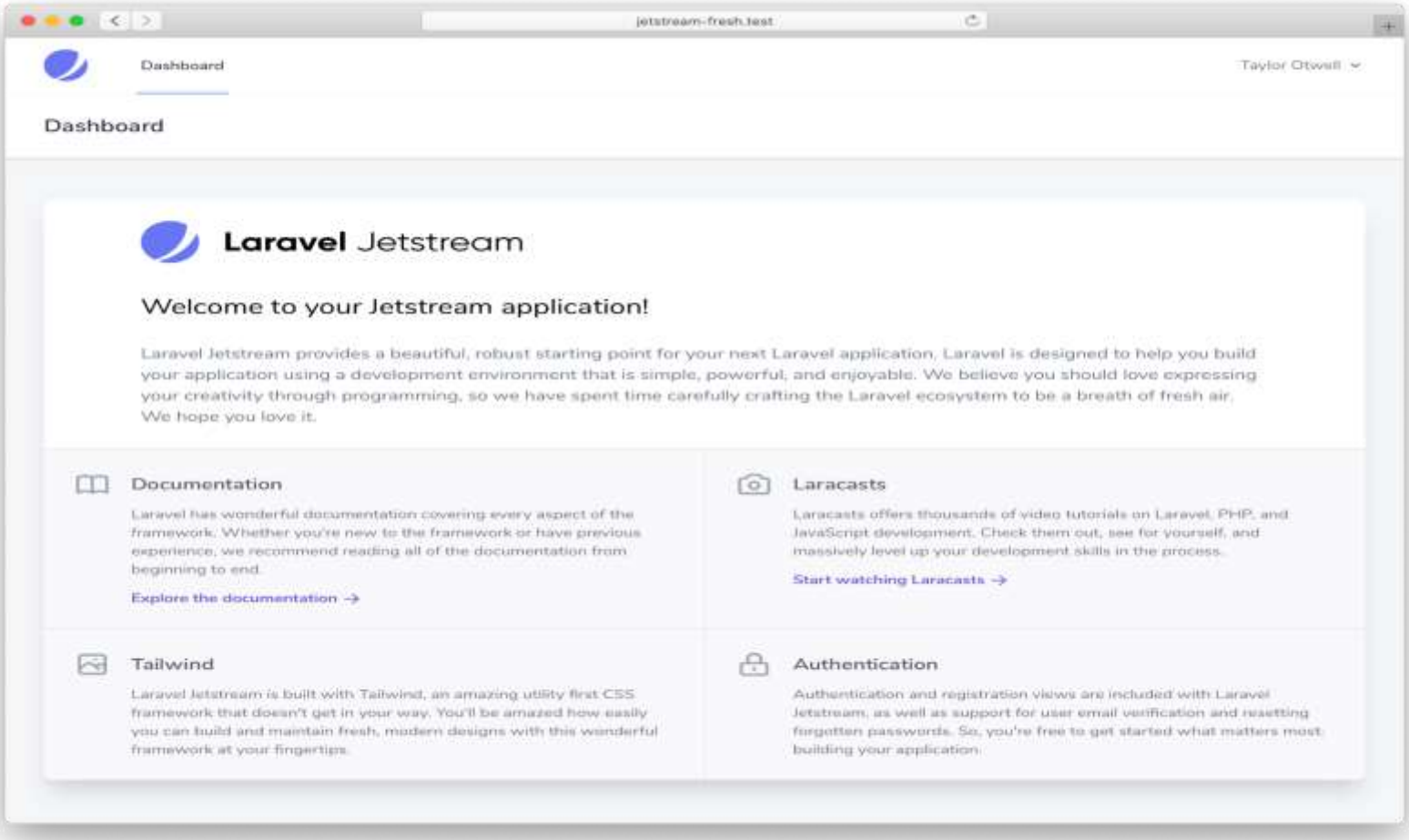
After Sail has been installed, you may run the sail:install Artisan command. This command will publish Sail's docker-compose.
yml file to the root of your application:

```
php artisan sail:install
```

Laravel Jetstream

Laravel Jetstream is a beautifully designed application starter kit for Laravel and provides the perfect starting point for your next Laravel application. Jetstream provides the implementation for your application's login, registration, email verification, two-factor authentication, session management, API via Laravel Sanctum, and optional team management features.
Jetstream is designed using Tailwind CSS and offers your choice of Livewire or Inertia scaffolding

Laravel Jetstream



Laravel Jetstream

Available Stacks

Laravel Jetstream offers your choice of two frontend stacks: Livewire and Inertia.js. Each stack provides a productive, powerful starting point for building your application; however, the stack you choose will depend on your preferred templating language.

Laravel Jetstream

Livewire + Blade

Laravel Livewire is a library that makes it simple to build modern, reactive, dynamic interfaces using Laravel Blade as your templating language.

This is a great stack to choose if you want to build an application that is dynamic and reactive, and is a great alternative to a full JavaScript framework like Vue.js.

When using Livewire, you may pick and choose which portions of your application will be a Livewire component, while the remainder of your application can be rendered as the traditional Blade templates you are used to.

Laravel Jetstream

Inertia + Vue

The Inertia stack provided by Jetstream uses Vue.js as its templating language. Building an Inertia application is a lot like building a typical Vue application; however, you will use Laravel's router instead of Vue router. Inertia is a small library that allows you to render single-file Vue components from your Laravel backend by providing the name of the component and the data that should be hydrated into that component's "props".

In other words, this stack gives you the full power of Vue.js without the complexity of client-side routing. You get to use the standard Laravel routing and view data hydration approaches that you are used to.

The Inertia stack is a great choice if you are comfortable with and enjoy using Vue.js as your templating language.

Daily Quiz

5. Where do we need to set database connection in Laravel?

- config.php
- setting.php
- In seed files
- **.ENV file**

6. hat is the minimum PHP version required to install Laravel 5.3?

- 7.1
- **5.6.4**
- 5.3.2
- 5.4.3

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- https://www.youtube.com/watch?v=ImtZ5yENzgE&ab_channel=freeCodeCamp.org
- https://www.youtube.com/watch?v=0yVDMcGp97g&list=PLjVLYmrlmjGfh2rwJjrmKNHzGxCZwBsqj&ab_channel=WsCubeTech
- https://laravel.com/docs/4.2/introduction#:~:text=Laravel%20attempts%20to%20take%20the,developer%20without%20sacrificing%20application%20functionality.
- https://www.tutorialspoint.com/laravel/laravel_overview.htm
- https://www.javatpoint.com/laravel

Weekly Assignment

- What are the default route files in Laravel?
- What is the Laravel Framework?
- What are available databases supported by Laravel?
- What is an artisan?
- How to define environment variables in Laravel?
- How to put Laravel applications in maintenance mode?
- What are the advantages of using the Laravel framework to build complex web applications?
- What is MVC architecture?
- How to create a route? Briefly describe with an example.

MCQ s

1. What is the use of .env file in Laravel?

- **Setting Up local Environment variable**
- Optimization
- Hosting
- None of these

2. Command to start Laravel Application?

- php artisan new
- php artisan
- php artisan start
- **php artisan serve**

MCQ s

3. Command to install laravel project?

- **composer create-project laravel/laravel myproject**
- composer new-project laravel/laravel myproject
- composer create-project new laravel/laravel myproject
- None of these

4. CLI Command to migrate in Laravel project?

- php artisan create migration
- php artisan migrate
- **php artisan serve**
- None of the above

MCQ s

5. Command to make migration in laravel?

- php artisan migration table
- **php artisan make:migration create_tags_table**
- php artisan make-migration digit
- None of these

6. Command to check the status of migration in laravel application?

- php artisan migration status
- php artisan status
- **php artisan migrate:status**
- None of these

Glossary Questions

1. ……create a controller in laravel by cmd?
- php artisan make: generate controller contoller_name
- php artisan make:controller generate
- **php artisan make:controller --plain**
- php artisan make:request controller_name create

2. View files in Laravel end in
- .blade.php
- .php
- .vue
- .blade

3. ……method returns the average value of a given key ?
- average()
- **avg()**
- median()
- avg_val()

4. Which of the following is true about Laravel?
- Laravel is an open-source PHP framework
- Laravel is robust
- Laravel is easy to understand
- All of the above

5. Laravel uses _____ to connect to an existing session and general-purpose cache
- Queues
- Cron
- Redis
- Command Bus

6. The _____ will run on every HTTP request of the application
- Global Middleware
- Route Middleware
- Both A and B
- None of the above

Expected Questions for University Exam

- What are the main features of Laravel?
- Explain routing and their types in details.
- How will you explain middleware in Laravel?
- What do you know about PHP artisan? Mention some artisan command.
- What are the differences between Laravel and Codeigniter?
- What are the major differences between Laravel 4 and Laravel 5.x?
- How do you make and use Middleware in Laravel?

Recap of unit

Students learnt:

- Basic Laravel
- Laravel Features
- Installation of Laravel
- Routing and their types
- Configuration of Laravel
- Laravel jetStream

Thank You