Laravel Tutorial

https://www.youtube.com/watch?v=-yFJh8CavLI

**Laravel** is an open-source PHP framework.

It uses the MVC architecture.

What is **Framework:** It is the collection of methods, classes, or files

**Architecture:** It is the specific design pattern that the framework follows.

* **M:** 'M' stands for **Model**. A model is a class that deals with a database.
* **V:** 'V' stands for **View**. A view is a class that deals with an HTML. Everything that we can see on the application.
* **C:** 'C' stands for **Controller**. A controller is the middle-man that deals with both model and view. A controller is the class that retrieves the data from the model and sends the data to the view class.

Installation of Laravel on Windows:

**Step 1: XAMPP Installation**

**Step 2: Composer Installation**

Composer is a dependency manager for a PHP programming

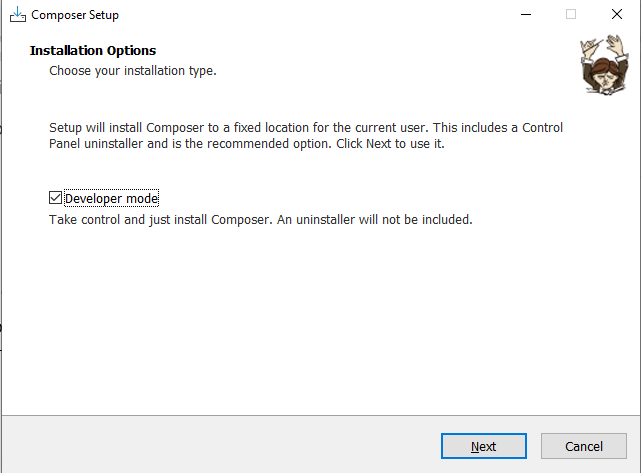
It is used to manages the dependencies of PHP software and required libraries

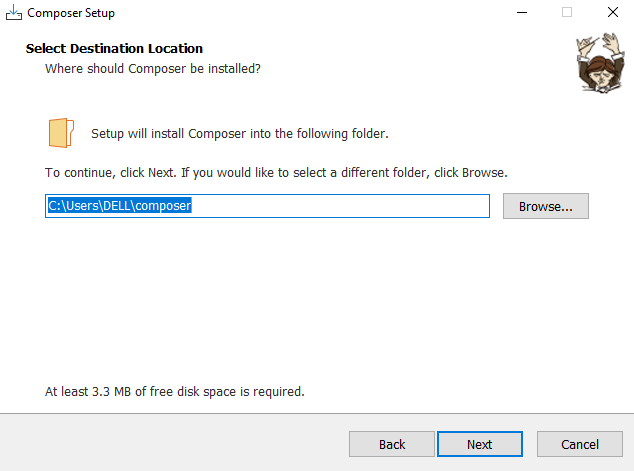
Composer runs through the **command line**.

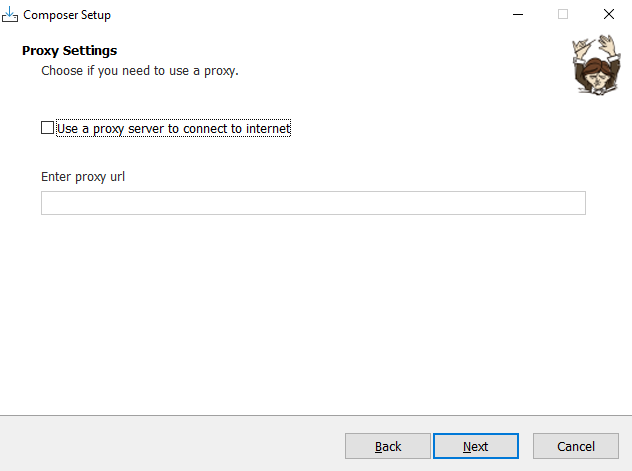
The main purpose of the composer is to install the  **libraries** for an application.

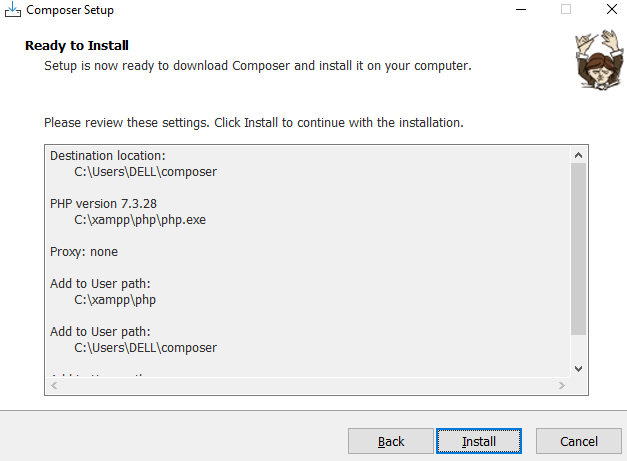
### **Steps to download the Composer:**

1. download the Composer: <https://getcomposer.org/download/>.
2. Once download click on the link **Composer-Setup.exe**.
3. Click on developer mode and say next

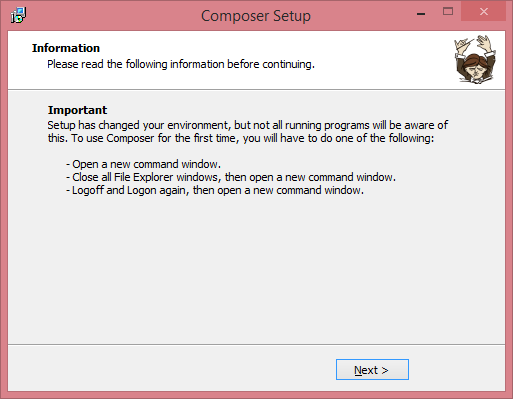


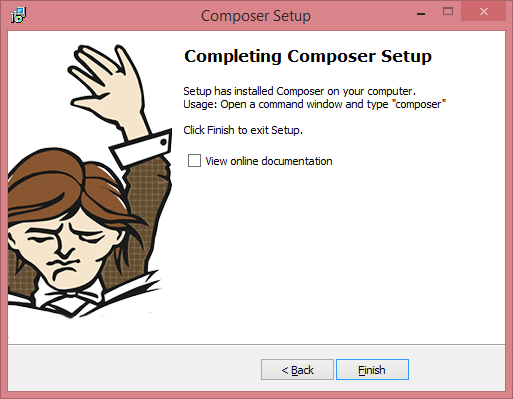


* Select the path of command-line PHP and the path is "***c:\wamp\bin\php\php5.6.40\php.exe***".
* Install Composer without a proxy server, so uncheck the box. Click on the **Next** button.
* 



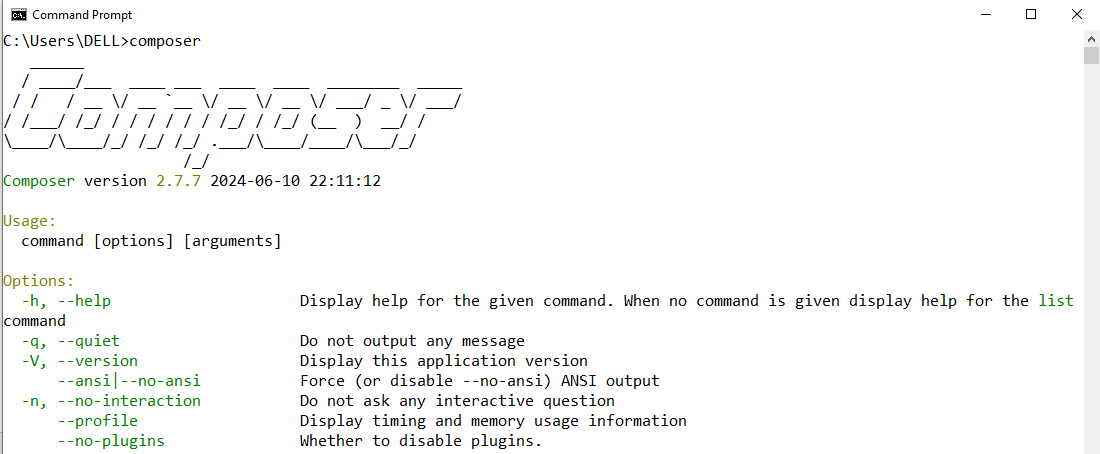
Click on the **install** button





The installation of the composer is completed.

To check whether composer is installed or not, open the **command prompt** and type **Composer**.



**Setp 3: Git Installation**

## **What is Git? :** Git is a free and open-source version control system

## **Steps to Install the Git**

* Click on the given link to download the Git: [ttps://git-scm.com/downloads](https://git-scm.com/downloads)

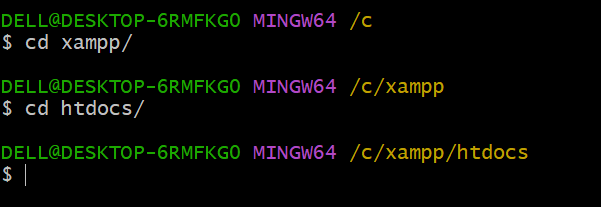
Go for windows version

Follow the steps : <https://www.javatpoint.com/laravel-git-installation>

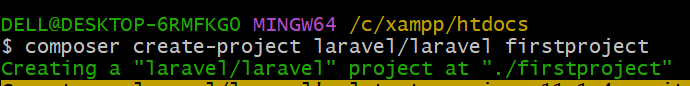
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Creating First Laravel Project

* Open the **Git Bash window**.
* Enter the ls command to know the current location.
* now we create a Laravel project in Xampp folder.
* In the above screen, **cd c:** command is used to move to the c drive and then we use the command **cd xampp/htdocs/** to move to the xampp folder.



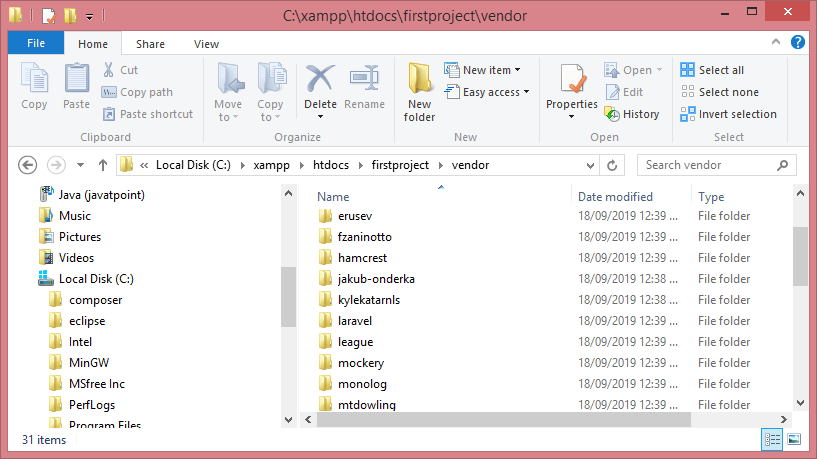
* Create a new laravel project.
* use the command **composer create-project laravel/laravel firstproject**.



**where,**

**laravel/laravel:** It is a vendor package.

**firstproject:** It is a project name.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Laravel Application Structure

# Link :

# https://www.youtube.com/watch?v=L7QxTLhsuVk

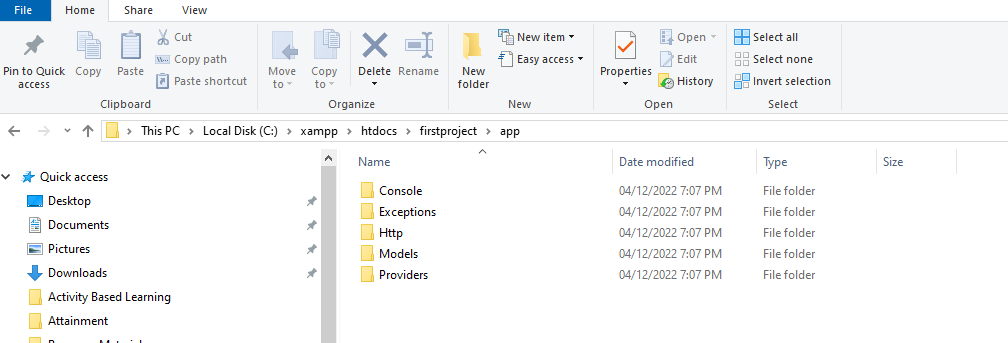
# <https://www.youtube.com/watch?v=P1vLmaZ2JLI>

# <https://www.youtube.com/watch?v=-yFJh8CavLI>

* The application structure is the structure of folders, sub-folders, and files available in the project

## **Laravel's app directory**

The app folder is one of the major folders in Laravel



**Console**

Console folder contains the artisan commands required for Laravel

**Exceptions**

Exceptions folder contains the various exception handlers.

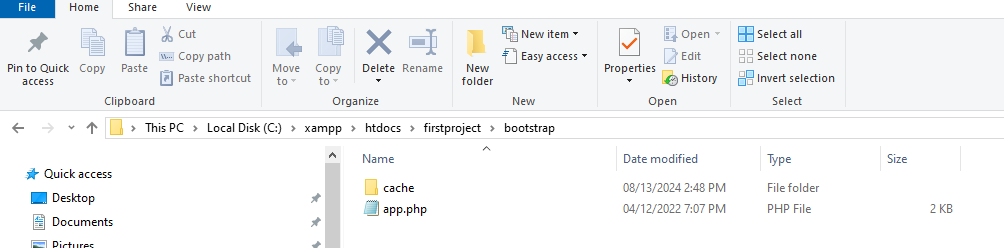
**Http**

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.

**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.

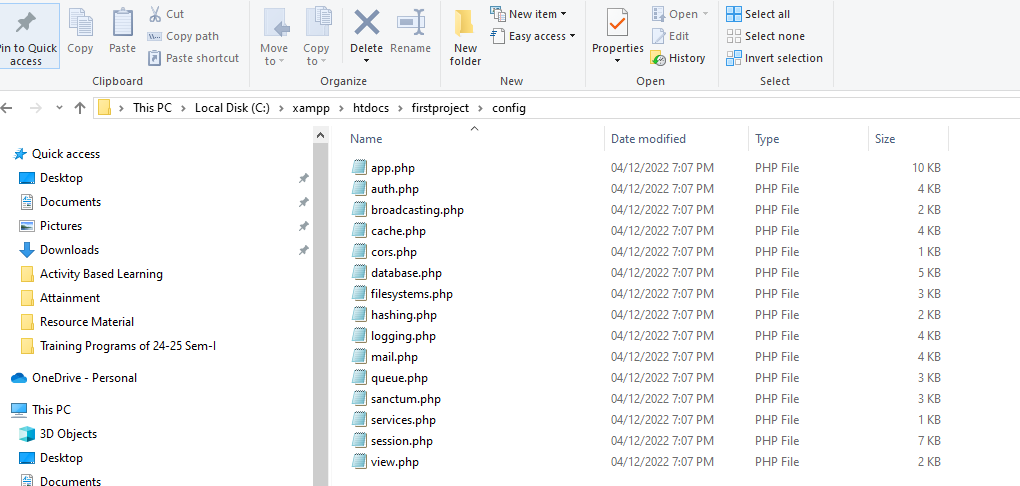
## **Laravel's bootstrap directory**



The bootstrap directory holds the files that are required to bootstrap the Laravel application and to configure auto-loading.

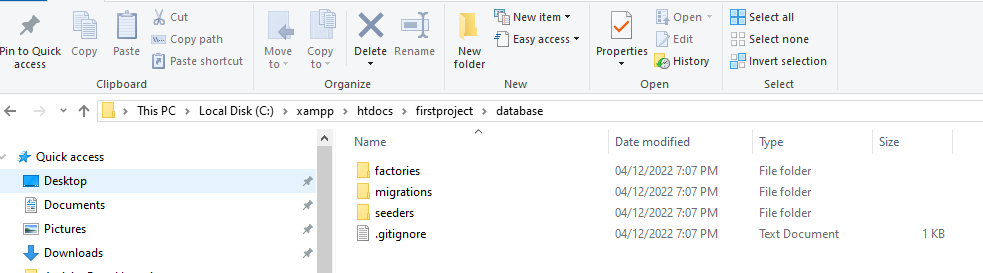
It contains the file **app.php** that initializes the scripts required for bootstrap.

## **Laravel's config directory**



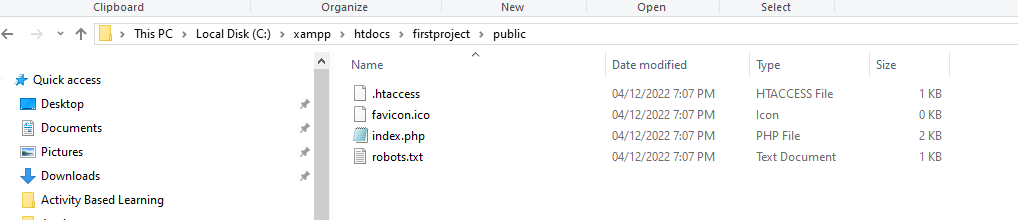
The config's directory contains the various configuration files required for the Laravel application

## **Laravel's Database directory**



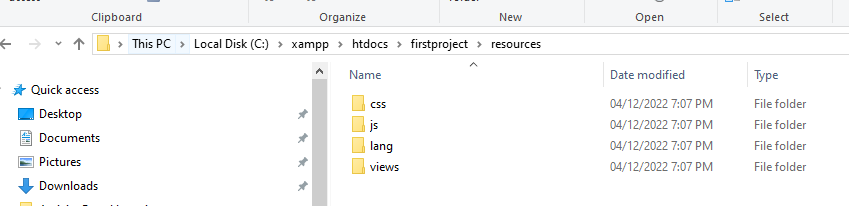
The database directory holds the database files

## **Laravel's Public directory**



The public directory is a root folder that contains the front controllers used for initializing the Laravel web application

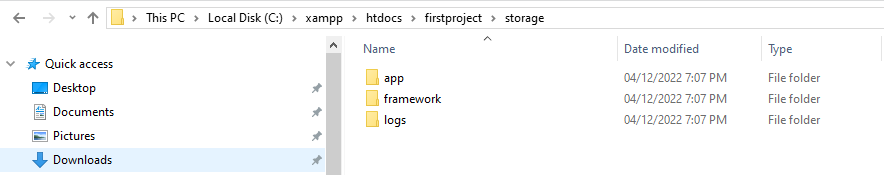
## **Laravel's Resource directory**



The resource directory holds the files used to enhance the web application

* **Assets:** This folder contains the files such as LESS and SCSS used for styling the web application.
* **Lang:** The lang folder provides configuration for localization or internationalization.
* **Views:** The views folder contains html files used to interact with the end-users, and it also plays an important role in MVC architecture.

## **Laravel's Storage Directory**



The storage directory contains those files which are required frequently when a Laravel project is running.

* **App:** It contains the files which are called frequently.
* **Framework:** It contains templates, filed based sessions, and file cache.
* **Logs:** This folder tracks the exceptions and error logs.

# Laravel Basic 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**

The **route/web.php** directory contains the definition of route files for your web interface.

Open file:

<?php

Route::get('/', **function** ()

 {

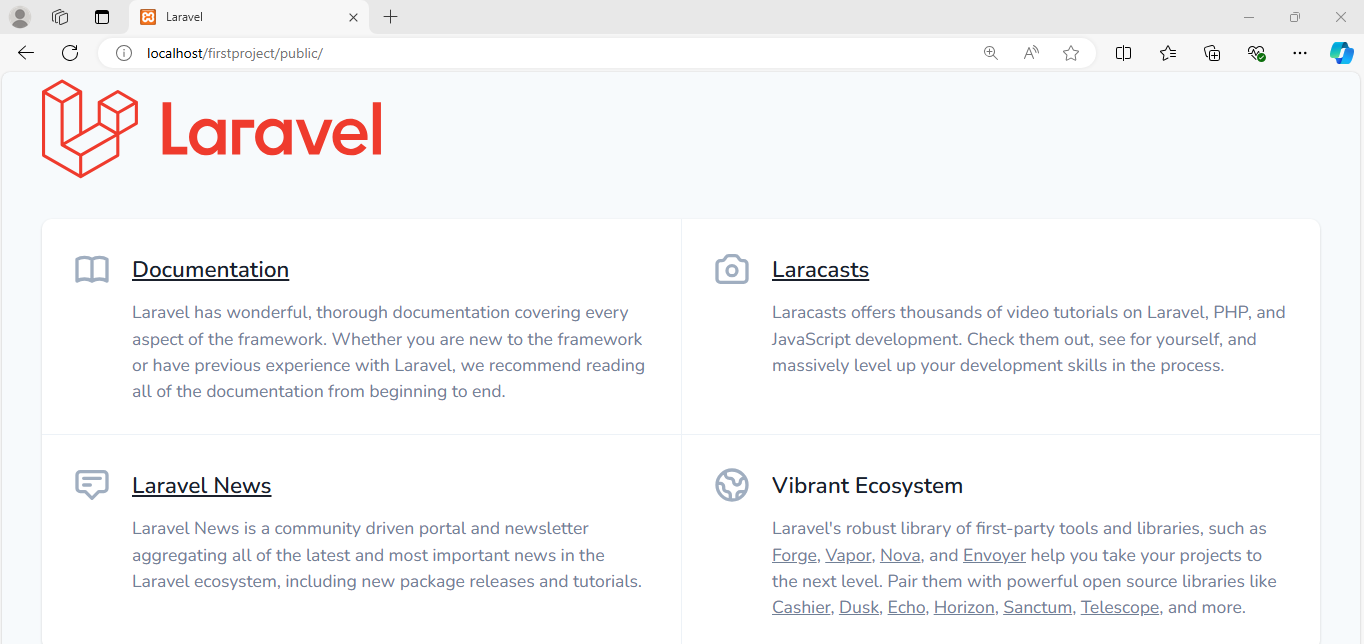
**return** view ('welcome');

});

Run :enter the **localhost/laravelproject/public URL** in the web browser.

 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.



Edit the file web.php in route folder of project

<?php

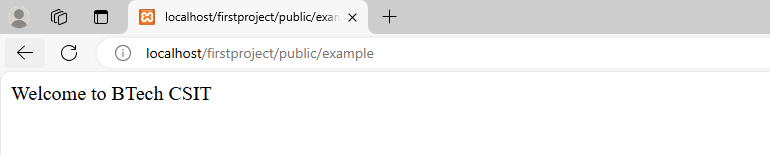
Route::get('/example', **function** ()

 {

**return** "Welcome to BTech CSIT";

});

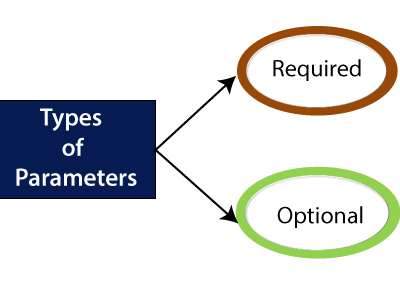
The route is defined in which URL is '/example', so we need to enter the URL "**localhost/laravelproject/public/example**" in the web browser.



Laravel Routing Parameters

There are two types of parameters we can use:

* Required Parameters
* Optional Parameters



## **Required Parameters**

The required parameters are the parameters that we pass in the URL.

**Examples without the route parameters.**

**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";**

**}**

**);**

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Example with route parameters:**

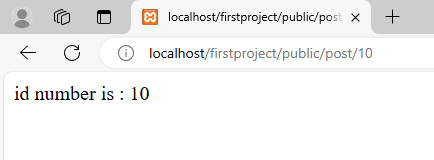
**Route::get('/post/{id}', function($id)**

**{**

**return "id number is : ". $id;**

**}**

**);**



**Route::get('/post/{id}/{name}', function($id,$name)**

**{**

**return "id number is : ". $id ." and the name is ".$name;**

**}**

**);**



## **Optional Parameters**

Route::get('college/{name?}', function ($name='RIT') {

return "The college Name is " .$name;

});

|  |  |
| --- | --- |
|  |  |

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

# Laravel Controllers

# What are the Controllers?

# Initially we were handling the request logic in the form of closures in route files; now, in place of using closures in route files, we use controller classes. Controllers are used to handle the request logic within the single class, and the controllers are defined in the "****app/http/Controllers****" directory.

# In MVC framework 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.

<?php

namespace App\Http\Controllers;

**use** Illuminate\Foundation\Bus\DispatchesJobs;

**use** Illuminate\Routing\Controller **as** BaseController;

**use** Illuminate\Foundation\Validation\ValidatesRequests;

**use** Illuminate\Foundation\Auth\Access\AuthorizesRequests;

**class** Controller **extends** BaseController

{

**use** AuthorizesRequests, DispatchesJobs, ValidatesRequests;

}

# In the above 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.

1. namespace App\Http\functions1;
2. namespace App\Http\functions2;

# 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.

# ****Create the controller through Git Bash Window:****

# Step 1: Open the Git Bash Window and type the command "php artisan make:Controller PostsController"

# 

# 

# The default code of PostsController.php is as follows

# <?php

# namespace App\Http\Controllers;

# use Illuminate\Http\Request;

# class PostsController extends Controller

# {

# //

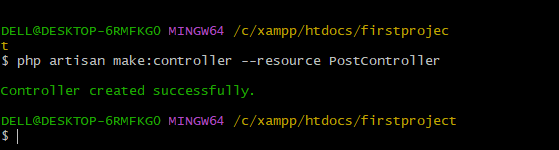
# }

The above 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.

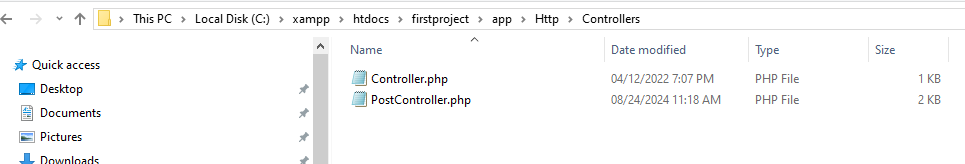
To create the Controller, we will first delete the **PostsController.php** from the project, which we have created in the previous step.

**Type the command:**

**php artisan make:controller -resource PostController**



move to your project to see whether the **PostController** file has been created or not



Now PostController.php file code is

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

//

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

//

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

//

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

//

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

//

}

}

**Here**

**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.

# Routing Controllers:

# Routing controllers allow you to create the controller classes with methods used to handle the requests

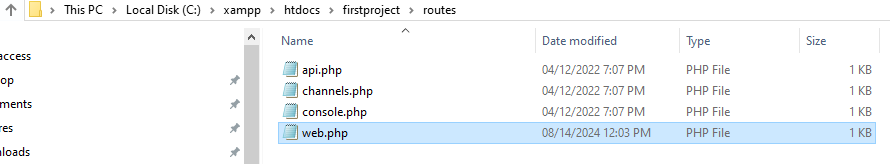
**Step 1:** First, we need to create a controller. We already created the controller named as '**PostController**' in the previous topic.

**Step 2:** Open the **web.php** file and write the following code:

Route::get('/post','PostController@index');

Or

Route::get('/post',[PostController::class,'index']);

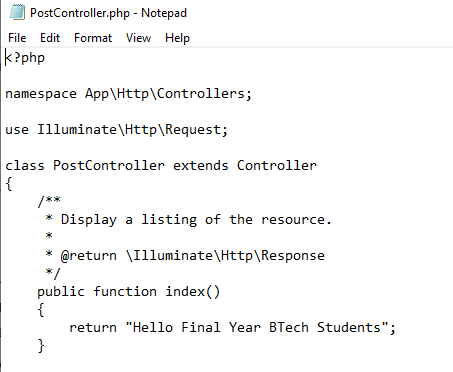


In the above code, '**/post**' is the URL that we want to access, and **PostController** is the name of the controller. The '**index**' is the name of the method available in the **PostController.php** file, and @index indicates that the index() method should be hit when we access the '/post' url.

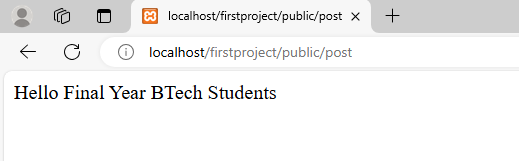


**Step 3:** Now open **PostController.php file and update the code of index function**

**return** "Hello Final Year BTech Students";



**Step 4:** Enter the URL in the browser, i.e., **localhost/laravelproject/public/host**, then the output would be shown as below:



# Passing data to controller:

**Step 1:** Open the **web.php** file and add the following code:

Route::get('/post/{id}','PostController@index');

Or

Route::get('/post/{id},[PostController::class,'index']);

# Change the index function in controller:

**public** **function** index($id)

{

**return** "ID is :". $id;

}

# IMP command if error:

# php artisan optimize

# php artisan serve

# 

# Laravel Views:

# What are the views?

# Views contain the html code required by your application. Views are located in the resources folder, and its path is resources/views.

**Simple example of views:**

**Step 1:** First, we create the view file named **Contact.php** in **resources/views** directory.

|  |  |
| --- | --- |
| **Create Contact.php in resources/views** | **Web.php**  **In public folder** |
| **<html>**  **<body>**  **<h1>Name of the Contact is: <?php echo $name; ?></h1>**  **</body>**  **</html>** | **Route::get('/contact', function(){**  **return view('Contact',['name'=>'Ashwini Patil , RIT']);**  **});** |

**Run by URL:** [localhost/firstproject/public/contact](http://localhost/firstproject/public/contact)



**another example of views through controller**

|  |  |  |
| --- | --- | --- |
| **Create aboutme.php in resources/views** | **PostController file** | **Web.php in In public folder** |
| <html>      <body>          <h1>About Us</h1>      </body>  </html> | **public function display()**  **{**  **return view('aboutme');**  **}** | **Route::get('/post',[PostController::class,'display']);** |

# Passing Data to Views:

**There are various ways of passing data to views:**

* By using the **name** array
* By using **with()** function
* By using **compact()** function

# Name Array:

# The name array is the array of data that is passed as the second parameter to the ****view()**** method.

# **Step 1:** First, we create the **student.blade.php**, which contains the view of the page.

# <html>

# <body>

# <h1>Name of the Students are : <br>

# <?php

# echo $name1;

# echo "<br>";

# echo $name2;

# echo "<br>";

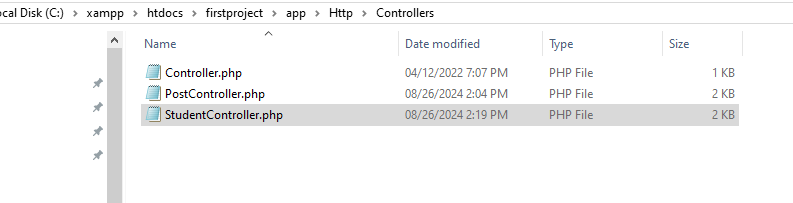
# echo $name3; ?></h1>

# </body>

# </html>

# **Step 2:** Now, we create the **StudentController.php** file

php artisan make:controller --resource StudentController



# In StudentController.php file

**public** **function** display()

  {

**return** view('student',['name1'=> 'Anisha','name2'=>'Nishka','name3'=>'Sumit']);

  }

# **Step 3: Now, we define the route in web.php file.**

Route::get('/details', 'StudentController@display');

Or

Route::get('/details', [StudentController::class,'display']);

# 

# With ()

# use the ****with()**** function to pass the data to views.

* First, we create the **student.blade.php** file which contains the view of the page.

<html>

 <body>

 <h1>Student id is :

 <?php

echo $id;

?>

</body>

</html>

* create the **StudentController.php** file

<?php

namespace App\Http\Controllers;

**use** Illuminate\Http\Request;

**class** StudentController **extends** Controller

{

**public** **function** display($id)

  {

**return** view('student')->with('id',$id);

  }

}

Define the route:

Route::get('/details/{id}', [StudentController::class,'display']);

OR

Route::get('/details/{id}', 'StudentController@display');

# Compact()

The compact() function is also used to pass the data to views. It contains a single parameter, i.e., the name of the variable.

* First, we create the **student.blade.php** file which contains the view of the page.

<html>

 <body>

 <h1>Name is :

 <?php

echo $name;?>

</body>

</html>

* create the **StudentController.php** file

 public function display($name)

  {

  return view('student', compact('name'));

  }

Add route in Web.php

Route::get('/details/{name}', [StudentController::class,'display']);



**Registration and login**

<https://magecomp.com/blog/laravel-10-custom-user-registration-login/>

**https://magecomp.com/blog/laravel-10-custom-user-registration-login/?srsltid=AfmBOooJrfzrmn8G9rzG\_9y\_hXWtFLFGL-C79a12yT7di7AnHoovq930**

**https://medium.com/@bbkgull/creation-of-user-login-and-registration-screens-with-laravel-8-8e563c7d5336**