**Laravel 7**

**Definition:** Laravel is a php framework. It has many build up package/function, which can be used in various purpose.

It follows **MVC pattern**. MVC means –

M - Model

V – View

C – Controller

Where Django follows **MVT pattern**. MVT means –

M – Model

V – View

T – Template

**Model:** Connect with Database.

**View:** Front-end side like which user see.

**Controller:** Where logic is implemented.

**Project installation:**

composer create-project --prefer-dist laravel/laravel foldername

**Start Local Development server:**

php artisan serve

**HTTP Request:** User request like as ‘get’, ‘post’, ‘petch’, ‘match’, ‘put’, ‘delect’ etc.

**Get Request Method:** It is used to retrieve all user data from server using specific URL.

**Post Request Method:** It is used to send data to server.

**Put Request Method:** It is used to update user data from server.

**Patch Request Method:** It is used to partially update data from server.

**Delete Request Method:** It is used to delete user data from server.

**Any Request Method:** It is used to done any things when URL is no defined.

**Match Request Method:** It is used to two or more request is done by one URL.

**Routes:** It means a specific path like URL.

Example –

localhost:8080/about

localhost:8080/contact

Basically we fixed our route in project’s route folded and it’s ‘web.php’ file.

**Route function define:**

Route::get('/contact',function(){

    return view('contactpage');

});

Here, if we search by “localhost:8080/contact” URL, we see the contactpage blade templated php file output.

contactpage is situated in ‘views’ folder named as contactpage.blade.php .

If you save any bladed php file into a ‘views’ subfolder. Then

Syntax –

return view(‘subfolder/contactpage’) ;

**Blade:** It is a template Where a variable is passing from one page to another page easily. Moreover It’s syntax is easy and short than non-blade file or normal php file.Moreover,By Blade file, We can write php and html tag in one page without using “<?php> ?>”

**Blade Template file Syntax:**

**For Condition:**

@if()

@else

@endif

Noted: @endif is must needed. But else is not must needed.

**Foreach Loop:**

@foreach

@endforeach

**Value print:**

{{ $var }}

**Valet & Homestead:**

That’s are virtual machine.where we can run php file without install php or XAMPP. Valet is needed for Mac. Homestead is needed for Windows and Linux.

**Middleware:**

Middleware used for apply condition in your website. Suppose, you can not go any page without login that rules are integrated by middleware.

**Create & Registered & Applied Middleware:**

**Step-1:** Create middleware in cmd in project folder.

php artisan make:middleware AgeMiddleware

**Step-2:** Registered Middleware.

Open ‘kernel.php’ file and insert in $routemiddleware function.

‘age’ => \App\Http\Middleware\AgeMiddleware::class,

**Step-3:** Condition set in handle function of AgeMiddleware file

if ($request->age <=200){

return redirect(‘home’);

}

**Step-4:** Define home route

Route::get(‘home’,function(){

echo “this is a home page”

});

**Step-5:** For which route is integrate the middleware

Route::get('/contact',function(){

    return view('contactpage');

})->middleware(‘age’);

**Step-6:** Now Check the middleware is working or not.

Use localhost:8080/contact?age=180

Show - 404 Not found

Use localhost:8080/contact?age=220

Show - Contact page successfully.

**Controller:**

Controller is a term which design a function and get details which happens when get HTTP request. That means, When We get a HTTP request(Get, Post etc) from user, It first comes to the routes. Routes check Controller’s method/function. Then that happens which are described in Controller’s method/function.

**Create Controller:**

Open cmd in project folder. Then types

php artisan make:controller controllername

Noted: Controller do not need to register in any file.

**Create Controller’s method/function:**

Open controllername file. Then types

Public function serviceMethod()

{

return view(‘servicepage’);

}

**Define Controller’s method/function:**

Route::get(‘/service’, ‘controllername@serviceMethod’);

**url,URL::to,Route:**

**url:** Go form one page to another page ,url is needed.

**Syntax:** <a href = “{{ url(‘service’) }}”> Service < /a>

**URL::to:** Go from one page to another page, URL::to is needed.

**Syntax:** <a href = “{{ URL::to(‘service’) }}”> Service < /a>

**Route:** Go from one page to another page, Route is needed. For changing url,Route is needed.

**Syntax:** <a href = “{{ route(‘service’) }}”> Service < /a>

In route file,

Route::get(‘/qeqw3e23eqee2e2e’, ‘controllername@serviceMethod’)->name(‘service’);

Url/Link: localhost:8080/ qeqw3e23eqee2e2e But show service page.

**Extends:**

Inherit one page to other extends is needed.

**Syntax:**

Type in index.blade.php file,

@extends(‘servicepage’)

**Yield:**

If change anywhere where extends is used.

**Step-1:** First type in servicepage.blade.php where you can change.

@yield(‘changecontent’)

**Step-2:** Then go to the index.blade.php and type

@extends(‘servicepage’)

@section(‘changecontent’)

<h1>Change the content </h1>

@endsection

**Asset:** Basically css,img,js etc file is Asset.

For this,css,img,js etc file must insert in ‘Public folder’ in project folder.

For insert asset file type

Link ref= “{{asset(‘https/wqr3r2’)}}”

OR, type

Img src = “{{URL::to(‘img/src/wrwwe.jpg’)}}”

**Project run by Xampp:**

Index.php and some file insert in root file and change asset link.

**Hosting without Xampp:**

Rename server.php as ‘index.php’.

**MySQL Database with Laravel 7**

In Laravel, MySQL Table is known as ‘Migration’.

**Database Create & Connection (Query Builder Way):**

**Step-1:** Go phpMyAdmin and create database as normal php mysql.

**Step-2:** Go .env file in your project file

Change DB\_DATABASE and type your database name which already created in phpMyAdmin

Example-

DB\_DATABASE=database\_name

**Create Table/Migration in project file & that table insert into phpMyAdmin**

**Step-1:** Open cmd in your project folder & type

php artisan make:migration create\_tablename\_table --create=tablename

Noted: Plural name of table is best practice, but not must needed.

**Step-2:** That table inserted into phpMyAdmin .So, open cmd in your project folder & type

php artisan migrate

**Change MySQL Table’s field or format**

**Step-1:** Open cmd in your project folder & type

php artisan migrate:rollback

**Step-2:** Edit table from your migrate file from project folder.

**Step-3:** Open cmd in your project folder & again type

php artisan migrate

**@csrf Token**

@csrf token use in html form. If anyone do not hack your website using harmful query, @crsf is used for this.

**Data/Value Insert in Database**

**Step-1:** First Design a form in your reg.blade.php file with ‘name’ attribute.

**Step-2:** Define name attribute as username,password etc.

**Step-3:** In Route,

Route::post(‘/regprocess’, ‘UserController@regMethod’);

**Step-4:** In UserController, Firstly define DB in heading part of UserController

Example-

use DB;

**Step-5:** In UserController, define regMethod

public function regMethod(Request $req)

{

$data=array();

$data[‘username’]=$req->username;

$data[‘password’]=$req->password;

//data\_check

//return response()->json($data);

$query = DB::table(‘tablename’)->Insert($data);

if($query){

return view(‘dashboadpage’);

}

else{

echo “Please try again !”;

}

}

**Validation**

In UserController’s userReg method,

  //validation password

        $validation\_data=$req->validate([

            'userpass' => 'requried|unique:registrations|min=8|max=12',

        ]);

**Validation Error Check:**

In login.blade.php file

@if ($errors->any())

<div class="alert alert-danger">

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

**Variable pass from one page to another page**

**Step-1:** In UserController file,

$query = DB::

if($query)

{

Return view(‘orderpage’,**compact(‘var1’, ‘var2’, ‘var3’)**);

}

Here compact() , is the main part to pass/throw variable.

**Step-2:** Catch the variable in order.blade.php file

Username : **{{$var1}}**

Email : **{{$var2}}**

Contact : **{{$var3}}**

Noted: Variable name must same in Controller file & Php file.

**Read All Data from Database**

**Step-1:** In controller page,

public function allOrder(){

$query = DB::table(‘orders’)->get();

Return view(‘allorderpage’,compact(‘query’))

}

**Step-2:** In orderpage.blade.php file,

<table>

<tr>

<td>username</td>

<td>email</td>

<td>contact</td></tr>

@foreach($query as @row)

<tr>

<td>$row->**username**</td>

<td>$row->**email**</td>

<td>$row->**contact\_no**</td>

</tr>

@endforeach

</table>

Here,**username,email,contact\_no** table field name.

**Read Only some data using Where Condition**

 public function subQuery(Request $req){

        $query=DB::table('orders')->where('username',$req->user\_name)->get();

        return view('querypage',compact('query'));

    }

**Read Only one data(which is first queried in table) using Where Condition**

 public function subQuery(Request $req){

        $query=DB::table('orders')->where('username',$req->user\_name)->first();

        return view('querypage',compact('query'));

    }

**Varibale/Value passing by URL**

**Step-1:** In link page file,

<a href= “{{ URL::to(‘order**/’.$row->id**)}}”>view</a>

**Step-2:** In route file,

Route::get(‘/order/{id}’, ‘UserController@subOrder’);

**Step-3:** In UserController file,

Public function subOrder($id){

$query=DB::table('orders')->where('username',$id)->first();

}

**Delete Data from Database**

**Step-1** && **Step-2** are same as **Varibale/Value passing by URL**

**Step-3:**

Public function deleteOrder($id){

$query=DB::table('orders')->where('username',$id)->delete();

return Redirect()->back();

}

Here, “return Redirect()->back()” is used for go to back page.

**Edit, Update Data from Database**

Step-1: First which data is already haven this shown in a html form.

For this, In allorder.blade.php file,

<a href="{{url('order/edit/'.$row->id)}}" class="btn btn-sm btn-info"> Edit</a>

Then In route file,

Route::get('/order/edit/{id}','UserController@editOrder');

Next, In UserController file,

 public function editOrder($id){

        $query=DB::table('orders')->where('id',$id)->get();

        return view('updateOrder',compact('query'));

    }

Then.In updateOrder.blade.php file,

     @foreach($query as $row)

                <form action="{{ url('/update/'.$row->id) }}" method="post">

               @csrf

                    <label for="fname">Username:</label><br>

                    <input type="text" id="fname" name="user\_name" value="{{ $row->username }}"><br>

                    <label for="fname">Order name:</label><br>

                    <input type="text" id="fname" name="order\_name" value="{{ $row->ordername }}"><br>

                    <label for="lname">Delievery Data:</label><br>

                    <input type="date" id="lname" name="d\_date" value="{{ $row->delievery\_date }}"><br><br>

                    <input type="submit" value="Update">

                    <input type="reset" value="Reset">

                </form>

                @endforeach

Now show this data which already inserted.

Step-2: Now update the data.

For this, In route file,

Route::post('/update/{id}','UserController@updateOrder');

Then, In UserController file,

 public function updateOrder(Request $req,$id){

        $data=array();

        $data['username']=$req->user\_name;

        $data['ordername']=$req->order\_name;

        $data['delievery\_date']=$req->d\_date;

        $query=DB::table('orders')->Where('id',$id)->update($data);

        if($query){

            echo"Update succesfully done !";

        }

        else{

            echo"You are alrady have this data";

        }

    }

**Image & TextArea upload in Database**

**Step-1:** In Order.balde.php file,

                <form action="{{ url('/orderprocess') }}" method="post" enctype="multipart/form-data">

               @csrf

                    <label for="fname">Username:</label><br>

                    <input type="text" id="fname" name="user\_name"><br>

                    <label for="fname">Order name:</label><br>

                    <input type="text" id="fname" name="order\_name"><br>

                    <label for="lname">Delievery Data:</label><br>

                    <input type="date" id="lname" name="d\_date"><br><br>

                    <label for="lname">Design file:</label><br>

                    <input type="file" id="lname" name="design\_file"><br><br>

                    <textarea rows="4" cols="50" placeholder="Details" name="details"></textarea><br><br>

                    <input type="submit" value="Submit">

                    <input type="reset" value="Reset">

                </form>

**Step-2:** In controller file,

 public function orderProccess(Request $req){

        $data = array();

        $data['username']=$req->user\_name;

        $data['ordername']=$req->order\_name;

        $data['delievery\_date']=$req->d\_date;

        $data['details']=$req->details;

        //Image upload

        $image=$req->design\_file;

        if($image){

            //Making Unique name for image

            $image\_name=hexdec(uniqid());

            //Get file extension like jpg,png etc

            $ext=strtolower($image->getClientOriginalExtension());

            //Join unique file name & file extention with dot like 12323.jpg

            $image\_full\_name=$image\_name.'.'.$ext;

            //First create a 'image' folder,then define the path where file is stored.

            $upload\_path='public/image/';

            //image url which insert in database

            $image\_url=$upload\_path.$image\_full\_name;

            //file upload in project folder

            $upload=$image->move($upload\_path,$image\_full\_name);

            //file url upload in database

            $data['image']=$image\_url;

            $var=$data['username'];

            $var1=$data['ordername'];

            $var2=$data['delievery\_date'];

            //return response()->json($data);

            $query = DB::table('orders')->Insert($data);

            if($query){

                return view('OrderComplete',compact('var','var1','var2'));

            }

            else{

                echo"Please try again !";

            }

        }

        else{

            $var=$data['username'];

            $var1=$data['ordername'];

            $var2=$data['delievery\_date'];

            //return response()->json($data);

            $query = DB::table('orders')->Insert($data);

            if($query){

                return view('OrderComplete',compact('var','var1','var2'));

            }

            else{

                echo"Please try again !";

            }

        }

    }

**Image & TextArea show**

<td><img src="{{ URL::to($row>image) }}" style="height:40 px; width: 70px;"></td>

<td>{{ $row->details }}</td>

**Database Relation**

Database Relation 3 types.

1. One to One
2. One to Many
3. Many to Many

**One to One Relation**

One to One Relation is One table’s one field related with another table’s just one field relation

**Requirements**

In two table, one same field which is unique and same data type, same length.

**Syntax**

In controller file,

public function joinOnetoOne($id)

{

$query = DB::table(‘orders’)->join(‘registrations’, ‘orders.username’, ‘registrations.username’)->select(‘orders.\*’, ‘registrations.userpass’)->where(‘orders.id’,$id)get();

return response()->json($query);

}

**Option/Select Attribute Update**

**Step-1:**

Get all data from database and get one data which match with ID in Controller file.

**Syntax:**

 $query=DB::table('orders')->where('id',$id)->get();

        $query2= DB::table('orders')->get();

        return view('updateOrder',compact('query','query2'));

**Step-2:**

In View file, First starting loop and show every option/select attribute data. Then which data is match with our id, that is selected by IF Condition.

**Syntax:**

<select name="order\_name">

                    @foreach($query2 as $row2)

                    <option value="{{ $row2>ordername }}"<?php if($row>ordername== $row2>ordername) echo "selected" ; ?> >{{ $row2->ordername }}</option>

                    @endforeach

 </select>

**TextArea Update:**

In view file, show which data is aleady haven in database.

      <textarea rows="4" cols="50" name="details">{{ $row->details }}</textarea><br><br>

Then other is same as text/option attribute.

**Image Update**

**Step-1:** First show old image in view file.

<label for="lname">Old Design:</label><br>

                    <img src="{{ URL::to($row->image) }}" style="height:40 px; width: 70px;"><br>

**Step-2:** Add Choose file input attribute add in view file.

 <label for="lname">Change Design:</label><br>

                    <input type="file" id="lname" name="design\_file"><br><br>

**Step-3:** Other is same as insert image mechanism. Oe.nly differ is where insert is situated. Just written update where insert is written and use Where condition in Controller file.

$image=$req->file('design\_file');

        if($image){

            //Making Unique name for image

            $image\_name=hexdec(uniqid());

            //Get file extension like jpg,png etc

            $ext=strtolower($image->getClientOriginalExtension());

            //Join unique file name & file extention with dot like 12323.jpg

            $image\_full\_name=$image\_name.'.'.$ext;

            //First create a 'image' folder,then define the path where file is stored.

            $upload\_path='public/image/';

            //image url which insert in database

            $image\_url=$upload\_path.$image\_full\_name;

            //file upload in project folder

            $upload=$image->move($upload\_path,$image\_full\_name);

            //file url upload in database

            $data['image']=$image\_url;

            //return response()->json($data);

            $query = DB::table('orders')->where('id',$id)->update($data);

            if($query){

                echo "Update Succefully done";

            }

            else{

                echo"You have same data already !";

            }

        }

        else{

            $query = DB::table('orders')->where('id',$id)->update($data);

            if($query){

                echo "Update Succefully done";

            }

            else{

                echo"You have same data already !";

            }

        }

**Delete Image**

Delete image is same as other.

 public function deleteOrder($id){

        $query=DB::table('orders')->where('id',$id)->delete();

        return Redirect()->back();

    }

Noted: By this, Image path is deleted from Database. But This Image is not deleted from my project folder. As a result, our project is overloaded.

For delete this image from my project folder, use “Unlink”

First get data use id. Then delete by Unlink

 public function deleteOrder($id){

        $query2 = DB::table('orders')->where('id',$id)->first();

        $image2 = $query2->image;

        $query=DB::table('orders')->where('id',$id)->delete();

        unlink($image2);

        return Redirect()->back();

    }

**Database Pagination**

Database pagination is a mechanism which helps to us data show partially in one page. That means, If we have 107 data and it show 10 or some data in per page. As a result, first page have 10 data. Second page have another 10 data and also third page have 10 data….

One page to another page is link up by 1,2,3…. as page number or Previous, Next

**Step-1:** If you link up one page to another page by 1,2,3… as page number, First go to Controller file, Then, using “paginate()” instead of “get()”

$query=DB::table('orders')->paginate(3);

Here, “3” means only 3 data show in per page.

**OR,**

If you link up one page to another page by previous,next, First go to Controller file, Then, using “paginate()” instead of “get()”

$query=DB::table('orders')->simplepaginate(3);

Here, “3” means only 3 data show in per page.

**Step-2:** Then use “{{ $query->links() }}” in view file after foreach loop…

@foreach($query as $row)

                <tr>

                <td>{{ $row->username }}</td>

                <td>{{ $row->ordername}}</td>

                <td>{{ $row->delievery\_date }}</td>

               <td> <a href= "{{ URL::to('/order/view/'.$row->username)}}">View</a>

                <a href="{{url('order/edit/'.$row->id)}}" class="btn btn-sm btn-info"> Edit</a>

                <a href="{{url('order/delete/'.$row->id)}}" class="btn btn-sm btn-danger"> Delete</a>

                <a href="{{ url('order/join/'.$row->id)}}" > join </a>

                 </td>

                </tr>

                @endforeach

                </table>

                {{ $query->links() }}

**Eloquent CRUD Operation**

For Eloquent CRUD Operation, create model.

php artisan make:model modelname

Noted: Model name is same as table name. Suppose if your table name “Students” , Model name is “Student”

In model,type…(For security purpose.As a result, anyone can not create database field from outside)

class modelname extends Model

{

    //For security. As a result anyone can not create database field from outside.

    protected $fillable = [

        'field1', 'field2', 'field3',

    ];

}

**Insert Data by Eloquent way**

**Step-1:** Define model name in controller file.So, type…

use App\d\_order;

**Step-2:**

 public function insertData(Request $req)

    {

        $orderObject = new modelname();

        $orderObject->username = $req->user\_name;

        $orderObject->project = $req->order\_name;

        $orderObject->d\_data = $req->d\_date;

       // return response()->json($orderObject);

      $query = $orderObject->save();

      if($query){

        return view('delieveryComplete');

      }

      else{

      }

    }

**Show all Data by Eloquent**

 public function dataShow()

    {

        $query=d\_order::get();

        return view('showDelievery',compact('query'));

    }

Noted: You can use “all()” instead of “get()”

**Show Individual Data by Eloquent**

public function individualDataShow($id)

    {

        $query=d\_order::find($id);

        return response()->json($query);

    }

Noted: You can use “findorfail()” instead of “find()”

**Delete Data by Eloquent**

 public function deleteData($id)

    {

        $query=d\_order::find($id);

        $query->delete();

    }

**Update Data by Eloquent**

Same as insert data

Public function updateData(Request $req,$id)

{

$data = modelname::find($id);

$data->username = $req->user\_name;

$data->project = $req->order\_name;

$data->d\_data =$req->d\_date;

$data->save();

}

**Resource Route**

Resource Route is a mechanism which define route & method automatically. That means When we create controller file with resource route , Controller file has already some method.

This methods are :

index() => Show index page which has a link to go form which stored data in database.

create() => Show the form.

store() => Stored the form data.

show() => Show the data from database.

edit() => Show data from database for editing.

update() => Update the data of database.

destroy() => Delete the data of database.

**Step-1:** In cmd, create controller with resource route

php artisan make:controller controllername –resource

**Step-2:** In cmd, check route list,

php artisan route:list

**Step-3:** In route list, check which method is attached with which route .Then define this.

**Database Relationship**

Three types Relationship

1. hasOne
2. hasMany
3. Belongsto

**hasOne & Belongsto Join**

hasOne join is One table join with another table(One to One join). That join is that where main table need only one data from another table. Example – Suppose main table has username, email etc but not has phone number. So, For get phone number, it join with another table which has phone number. That join is “hasOne join”. The reverse of hasOne join is “Belongsto join” In Belongsto join, main table has only one data, but retrieve other data from another table.

**hasOne join by Query Builder way**

  // hasOne join proccess

    public function contactPage()

    {

        $query = DB::table('users')->join('phones','users.id','phones.userid')->select('users.username','phones.\*')->get();

        return response()->json($query);

    }

**Belongsto Join by Query Builder way**

public function contactPage()

    {

        $query = DB::table('phones')->join('users','phones.userid','users.id')->select('users.username','phones.\*')->get();

        return response()->json($query);

    }

Here, Select operation is needed to collect some data only from two tables.

**hasOne join by Eloquent way**

First create two model. Then One model has define another model. So, In main model, type…

public function phone()

    {

        return $this->hasOne('App\Phone');

    }

Then Controller same as display date from database.

In view file, type…

$row->phone->phone\_number

**Belongsto Join by Eloquent way**

public function phone()

    {

        return $this->belonsTo('App\Phone');

    }

Then Controller same as display date from database.

In view file, type…

$row->phone->phone\_number

**Three table join by hasOne / belongsto by Query Builder Way**

In Controller page, type…

  // hasOne join proccess

    public function contactPage()

    {

        $query = DB::table('users')->join('phones','users.id','phones.userid')

        ->join('posts','users.id','posts.phone\_id')

        ->select('users.username','phones.\*','posts.\*')

        ->get();

        return response()->json($query);

    }

**Three table join by hasOne / belongsto by Eloquent Way**

In main model, type…

 public function phone()

    {

        return $this->hasOne('App\Phone');

    }

    public function post()

    {

        return $this->hasOne('App\Phone');

    }

**Authentication**

Laravel has default authentication system. Authentication means login, register, forget password in any site. For Authentication install, type in cmd of project folder…

composer require laravel/ui

Then again type in cmd…

php artisan ui vue --auth

**Css/js etc file link up with login, register, forget password page (For styling this pages)**

**Step-1:** Install npm(node package module). So, type in cmd of project folder…

npm install

**Step-2:** Run npm’s package. So, type again in cmd…

npm run dev

**For working Authentication**

**Step-1:** Type in App\Providers\AppServiceProvider file’s “boot” function,

public function boot()

    {

        schema::defaultStringLength(191);

    }

**Step-2:** Type in App\Providers\AppServiceProvider file,

use Illuminate\Support\Facades\Schema;

**Use User data after user login**

{{ Auth::user()->name }}

{{ Auth::user()->email }}

**Registration off & Only login show(use in Admin Panel)**

In route file

Type into the Aute::Route()…

['register' => false]

That means type…

Auth::routes(['register' => false]);

**Change Password in Authentication**

In controller file, first use…

use Illuminate\Support\facades\Auth;

use Hash;

use DB;

Then,

public function changePassword(Request $req)

    {

        $data = array();

        $o\_password = Auth::user()->password;

        if(Hash::check($req->old\_password,$o\_password)){

                $userid = Auth::user()->id;

                $req->password = Hash::make($req->password);

                $data['password'] = $req->password;

                $query = DB::table('users')->where('id',$userid)->update($data);

                if($query){

                    echo"password change successfully!";

                }

        }

        else{

            echo"Please enter right password";

        }

    }

**Collective Form in Laravel**

Collective form is more secure than normal html form tag.

**Install Collective Form**

Open cmd in project folder and type…

$ composer require laravelcollective/html

**Collective Form Syntax**

//form start tag

{!! Form::open(['url' => '/login', 'method' => 'post']) !!}

//write any input tag

{!! Form::close() !!}

//form end tag