Junait Islam Shipon
01885655997

Larcavel Fundamental

Verson 9.x

junait Islam Shipon
web developerc
01885655997

Tools For Laravel

1. Composer
2. Xampp
3. Visual studio code

Install serial

1. Xampp
2. Composer

Terminal: composer create-project (--prefer-dist) laravel/laravel blog

laravel project কে run করার জন্য নিশ্চিত হতে হবে যে project ফোল্ডার সিলেক্টেড আছে কি না।

Terminal: cd blog

php artisan serve [Laravel development server started]

project structure

App → Http → middleware

bootstrap → css framework no

config →

database → যেটা ডেটার টেবিল আপডেট বা ডেটাবেজে কাজ করার জন্য database/migrations এর মাধ্যমে করতে হয়।

public → index.php [main file]

robots.txt [search Engine কাজ / ধরতে পারবে কি না]

resources → JS / CSS / SASS ফাইল থাকে

view / [আমরা যত কিছু দেখতে পাই সব ফাইল এখানে]

vendor → বিভিন্ন লাইব্রেরি থাকে [কখনো Edit করার প্রয়োজন হবে না]

.env [environment variable] file [DB Details]

composer.json → info

MVC - পদ্ধতি

Three part of Application বা mvc হলো সফটওয়্যার তৈরি করার একটি পদ্ধতি, যেখানে পুরো software কে তিনভাগে ভাগ করা হয়েছে বা source কোডকে সাজানো হয়।

1st part → Model [data structure]

2nd part → view [The visible part / frontin part]

3rd part → controller [Logics of application / back-end]

Data Insert → view → controller → model →

Essential PHP Artisan Command

Command name	Action
Php artisan serve	→ To run laravel application
php artisan make: controller ControllerName	→ To make new controller
php artisan make: Model ModelName	→ To make new model
php artisan make: Middleware MiddlewareName	→ To make new m
php artisan make: migration tablemame	→ To make migration file
php artisan migrate	→ To run migration
php artisan ~~case~~ cache: clean	
" route: "	
" config: "	
" view "	

laravel Basic Routing

Routing is a process to set url, that requested by browser.
route/web.php directory contains the definition of route files.

নিয়ন্ত্রণ করবে callback function

```
Route::get('/hello', function(){
    return "Hello world";});
```

Laravel Basic Routing with View

প্রথমে resources ফোল্ডারের মধ্যে views তৈরি করতে হবে .Blade.php

Laravel Basic Routing with Controller

* First Create Controller php artisan make: controller ControllerName
* Then route with controller name and method.

siteController.php

```
{
  function Home () {
        return "I am from site controller Home method";
        }
```

web.php

```
Route:: get ('/', 'APP/Http/controllers/siteController@Home');
```

Laravel Basic routing with controller and view

view → controller → route

siteController.php

```
{
  function Home() {
        return view ('HomePage');
```

web.php

no change

HomePage.blade.php

```
<h1> —— </h1>
```

Routing with href link

```
<a href="{{ url('/') }}" > Home </a>
```

Note: / দেওয়া যাবে না।

Routing parameter

web.php

```
Route::get('/Name/{myName}', ...../siteController@demo');
```

siteController.php

```
function demo ($myName){
    return $myName; }
```

Note: we can use multiple parameter.

Routing parameter with view

site controller.php

```
function demo ($myName, $age); {
return view ('parameter', ['firstKey'=> $MyName, ...]);
}
```

parameter.blade.php

```
{{ $firstKey }}
```

For loop → blade view

```
@for ($i=0; $i<20; $i++)
    body
@endfor;
```

Laravel Routing group

```
Route::prefix('account') -> group (function () {
    Route::get ('/profile', function () {
        return 'profile';
    });
    ⋮
});
```

laraval Controller structure

দাদা Base controller

↓

বাবা Controller

↓

নাতি/পুতি/আপনি Those we create

Single Action Controller

Single Action Controller - যেটি একটি মাত্র কাজ করে।

MyController.php

```
public function space __invoke () {
    return "_____";}
```

Resource Controller

Resource controller – যা সব ধরনের Http request Handle করে।

1. index ()
2. create ()
3. store (Request $request)
4. show ($id)
5. edit ($id)
6. update (Request $request, $id)
7. destroy ($id)

```
Route::resource('photos', 'photoController');
```

Verb	Url	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
PUT/PATCH	/photos {photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

For create resource controller!

```
php artisan make:controller photoController --resource
```

Blade statement is used to write php inside html view

- {{....write php here.....}} → Filter by html specialchars function and prevent xss attacks
- {!!...write php here..!!} → No

Foreach loop

sitecontroller.php

```
function test () {
    $hi = array("__");
    return view('home', ['hi'=>$hi]);
```

home.blade.php

```
@foreach ($hi as $hx)
    {{ $hx }}
@endforeach
```

forelse loop

```
@forelse ($x as $y)
    body
@empty
    <p> No data </p>
@endforelse
```

IF, ELSE condition inside blade view

```
@if ($login_status_key == true)
    <h1> I am login </h1>
@elseif ($login_status_key == false)
    <h1> I am not login </h1>

@else
  <h1> not found </h1>

@endif
```

loop special properties

property	Discription
$loop->index	The index of the current loop iteration
$loop->iteration	The current loop iteration (starts at 1)
$loop->remaining	The iterations remaining in the loop.
$loop->count	The total number of items in the array
$loop->first	This is the first iteration through the loop
$loop->last	This is the last iteration through the loop
$loop->even	This is an even " " " "
$loop->odd	This is an odd " " " "
$loop->depth	The nesting level of the current loop
$loop->parent	When in a nested loop, the parent's loop variable.

```
@break

@if ($loop->even)
    {{ }}
@endif
```

PHP Inside Blade View

```
@php
    echo "Hello"
@endphp
```

Including Subview | Blade view

```
@include ('path')
```

passing data to sub view

Including subview Advance

```
@includeif ('~~menu~~subview.menu')
```

OR

```
@includeif ('subview/menu')
@includefirst ('subview.footer', 'subview.menu']) 
@includewhen ($boolean, 'sbview.footer', ['some' => 'data'])
@includeunless
```

use master layout

```
@extends('layout.masterlayout)
```

Blade template Inheritance

```
@yield ()
@section ()
```

Master layout

Home.blade.php

```
@section('content')
<botton> L </butoon>
⋮
@endsection
```

masterlayout.blade.php

```
@yield('content')
```

{{asset ()}}

@extends('layout.app')

create New Migration

Php artisan make: migration Migration Name

up method

1. works only at front Migration
2. can Modify table, Column, Index create table, Column, Index and also delete theses etc

down method

1. works only For Migration Roll back
2. Same

create table

```
Schema::create('students', function (Blueprint $table) {
    $table->bigIncrements('Id');
    $table->string('name');
    $table->string('roll');
    $table->timestamps();
});
```

```
$table->string('name')->nullable();
```

```
php artisan migrate
```

Blueprint Method

1. bigIncrements('ID'); → Auto Increment with primary key.
2. bigInteger('votes'); → big number
3. $table->binary('data'); → Image etc / binary large image
4. boolean('confirmed'); → yes or No
5. char('Name', 50); → 50 charter only
6. date('created_at');
7. dateTime('created_at');
8. dateTimeTz('created_at');
9. ipAddress('visitor');
10. longtext('description') → long Text
~~11. string('name',100)~~
11. string('city', 100); → Higest 100
12. timestamps(); create at, updated at
13. year('...');

Column Modifiers

```
$table->string('name', 50)->change() → change size
renameColumn('from', 'to');
dropColumn('votes');
```

required Dependencies composer require doctrine/dbal

```
public function up()
{
Schema::table('student', function (Blueprint $table) {
$table->string('Roll', 50)->change();
});
}
```

Delete table / rename

```
public function up()
```

1. `Schema::rename('from', 'to');`
2. ——————→ `::drop(——);`
3. ——————→ `::dropIfExists('—');`

Migration Roll Back

php artisan migrate:rollback → rolls back the last batch
migrate:rollback --step=2 → rolls back last 5
migrate:reset → rolls back all
migrate:refresh → " and creat new

down() method

61

back side/rool back

Raw SQL Queries

use Illuminate/Support/Facades/DB;

Data show

function xxx() {
return DB:select('____');

x ——— x ——— x ——— x ——— x

web.php

```
Route::get('/', 'App\Http\Controllers\siteController@dataShow');
```

siteController.php

```
use Illuminate\support\Facades\DB;
{
  function dataShow () {
        $jsonData = DB::select('————');
        $json_st = json_encode($jsonData);
        $select_Data = json_decode($json_st);
        return view('show', ['selectKey' => $select_Data]);
}       }
```

show.blade.php

```
@foreach ($selectKey as $item)
    <tr>
        <td> {{$item -> Name}} </td>
@endforeach
```

Data Insert

1. Download axios from github.com/axios/axios
2. Include it app.blade.php
3. Name : <Input type="text" Id = "name">

<Button> Submit </button>

onclick = "senddata()"

```
function senddata(){
    var name = document.getElementById('name').value;

    alert(name);
}
```

axios → source code download করে লোড করতে হবে

Insert Data into DB by axios এ্যাক্সিওস

বিশেষ নোট: অতিরিক্ত স্পেস ও স্পেস নাম বিভিন্ন জায়গায় ব্যবহার অনুসরণীয়।

web.php

```
Route: get('/', 'App/Http/Controllers/SiteController@insert_data');
Route::post('/insertData', 'App/---------@inserting-data');
```

siteController.php

```
function inserting-data(Request $req){
    $name = $req->input('myname');
    $roll = $req->input('myroll');
    $result = DB::insert('INSERT INTO ------('Name', 'Roll') values
            (?,?)', [$name, $roll]);
    if ($result){
        return "successful";
    }
}
}
```

insert.blade.php

```
<input type="text" placeholder="Name" id="name">
<script src="{{asset('js/axios.min.js')}}"> </script>
<script src="{{asset('js/custom.js')}}"> </script>
<button onclick="sendData()"> Insert </Button>
```

P.T.O.

custom.js

```
function sendData (){
    var name = document.getElementById ('name'). value ;
    "
    var url= "/insertData";
    var data = {myname: name, myroll: roll };

axios.post (url, data)
    .then (function (response){
      alert (response. data);
    })
    .catch (function (err){
      alert ("Error");
    }
```

Delete_Data

web.php

```
Route: get ('/', App /Http/ controllers /site controllers@deleteData);
"  post ('/deleteData', ———— @deletingdata');
```

sitecontroller

```
function deletedata () {
return view ('Delete'); }
function deletingdata (Request $req) {
$id = $req -> input ('myid');
$result = DB:delete ('___ID = ?', [$id]);
if ($result){
    return "successful";
}else{
    return " Failed";
}
}
```

custom.js

```
function sendData () {
var dlt_id = document.getElementById ('__').value;
var url = "/deleteData";
var data = {myid: dlt_id};
```

axios.post → Same as insert

Delete.php

Same as insert

* UPDATE *

Laravel Query Builder

Retrieving Results

1. All Rows From A table → DB::table('—')->get();
2. Select A single Row → DB::table('—')->Where('name', '=', 'junait')
3. নির্দিষ্ট ID নম্বর সিলেক্ট করা → DB::table('student')->find(8);->first();
4. নির্দিষ্ট কলাম সিলেক্ট করা → DB::table('student')->pluck('Name', 'Roll')

Selects

1. All rows From A table → DB::table('—')->distinct()->get();
2. Select a column → DB::table('—')->Select('—')->get();
3. " multiple " DB::table('—')->Select('—', '—')->get();

Merge

```
function mergedata() {
    $first = DB::table('student')->distinct()->get();
    $second = DB::table('school')->  "  "
    $total = $first->merge($second);
    return $total;
}
```

MODEL

model properties

protected $table = 'table name'; → To define table name.

" $primaryKey = 'id'; → To define primary column

" $incrementing = false ; → To define primary column Data type

" $KeyType = 'int' → To define increment status.

" $timestamps = false → " timestamps "

public (all)

---

sitecontroller.php

```
use App/Models/modelName;

function selectall () {
    $result = student::get ();
              ↓
            ModelName
    return $result; }
```

File upload

Home.blade.php

```
<input type="file" id="file_upload">
<button onclick="onupload()"> upload </button>
```

(axios লেখা থাকতে হবে)

javascript

```
function onupload(){
let myFiles = document.getElementById("file_upload").files[0];
let fileName = myFiles.name;
let fileSize = myFiles.size;
let fileFormat = fileName.split('.').pop();
let fileData = new FormData();
fileData.append('Filekey', myFile);
let config = {headers: {'content-type': 'multipart/form-data'}}
let url = '/fileup';
axios.post(url, fileData, config)
.then(function (response) {
}).catch(function (error) {

})
}
```

upload - controller

```
function onfile (Request $request) {
    $request->file (key: 'filkey')->store (path: 'img');
```

Route

```
Route: post (uri: '/fileup', action: 'uploadController@ ....');
```

App/Http/controllers/

File upload with success message

Home.blade.php

```
<input type="file" id="upload_id">
<button onclick="onupload ()" id="upload_btn> </b>
```

web.php

```
Route: post (uri: '/upload', action: 'App/Http/controllers
/siteController@onupload
);
```

custom.js

```
function onupload () {
let Myfile = document.getElementById ('upload_id').files[0];
let fileName = Myfile.name;
let fileSize = Myfile.size;
let fileData = new FormData();
fileData.append ('FileKey', Myfile);

let config = {headers: {'content-type': 'multipart/form-data'}}
let url = '/upload';

axios.post (url, fileData, config)
.then (function (response){
$ ('#upload_btn').html ('upload success');
setTimeout (function () {
$ ('#upload_btn').html ('upload');
}, 3000)
}).catch (function (error) {
})
}
```

$ ('#upload_btn').html (spiner)

siteController.php

```
function onupload (Request $request){
$result = $request->file (key: 'FileKey')->store(path: 'img');
return $result;
}
```

File upload with percentage

↓

see the zip file

File upload with MB

↓

see the zip file

public Disk upload

controller

↓

```
function onfileup (Request $request) {
    $path = $request->file(key: 'Filekey')->store(path: 'public')
}
```

```
php artisan storage:link
```

```
<img src="{{asset('storage/image rename')}}">
```

Laravel Middleware class

Middle + ware → মধ্য + সতর্ক

Kernel.php →

RouteMiddleware এর মধ্যে নতুন করে middleware অ্যাড করে দিতে হবে [সেটা তৈরি করা হয়েছে]

kernel.php
↑
class name
↑

web.php

```
Route::get('/', .....)->middleware('login');
```

---

web.php

```
Route::get('/{token}', .....)->middleware('login');
```

login.php

```
$login = $request->token
if($token == "xxx") {
return $next($request);
} else {
return ("user blocked");
}}
```

Session

Store

```
$request->session()->put('name', 'junait');
```

Retrive

```
return $request->session()->get('name');
```

Clear

```
$request->session()->forget('name');
$request->session()->flush();
```

Cookies

use cookie

```
use Illuminate/support/Facades/cookie;
```

Store

```
cookie::queue('namekey', 'Junait', 4000);
```

Retrive

```
~~return $request~~
return cookie::get('Namekey');
```

Clear

```
cookie::queue(cookie::forget('namekey'));
```

Database: Seeding

seeding is a process to insert some dummy data in table

- php artisan make:seeder UserSeeder
- composer dump-autoload
- php artisan db:seed --class = UserSeeder.

```
public function run ()
{
  DB::table ('students') -> insert ([
      'name' => Str :: random (10),
      'email' => Str: random (10). '@gmail.com',
      'Password' => Hash: make ('password') ;
  ]);
```

total_seeder.php

```
public function run ()
  {
    $this-> call([
          StudentSeeder :: class,
          ExamSeeder :: class,
    ]);
  }
```

Most useable command

```
php artisan serve

php artisan serve --port=8080

php artisan migrate

php artisan make:migration

php artisan make:model

php artisan storage:link

php artisan db:seed --class=junait

php artisan migrate:fresh --seed

php artisan optimize

php artisan optimize:clear

php artisan route:cache

php artisan view:clear

php artisan make:model modelName

php artisan make:controller UserController
```

Advance Laravel Commands

```
php artisan list

php artisan --version

php artisan env

php artisan route:list


php artisan make:seeder EduCBASeeder

php artisan make:middleware Middleware_Name

php artisan make:mail Course_Enrolled@EduCBA

php artisan make:event EduCBA_Analytics_Enrolled

php artisan make:model Project --migration --controller --resource

php artisan make:provider OurServiceProvider

php -v
```