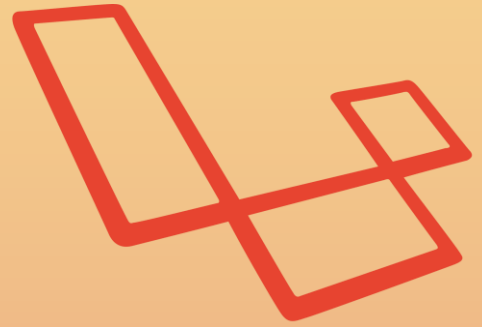


Laravel

การพัฒนา Web Application ด้วย Laravel Framework

CONTENTS

Laravel Framework

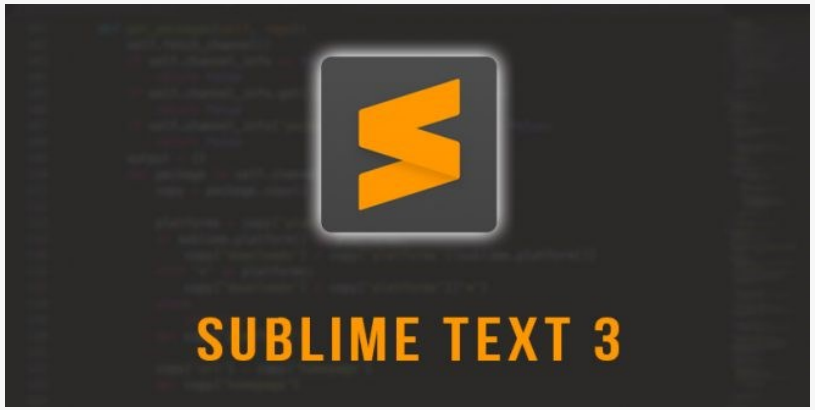
01.

Setup Environment & Laravel Project

การติดตั้งโปรแกรมจำลองเซิร์ฟเวอร์ และสร้างโปรเจ็คใหม่

Xampp เป็นโปรแกรม Apache web server จำลอง web server เพื่อทดสอบ สคริปหรือเว็บไซต์ในเครื่องของเรา โดยที่ไม่ต้องเชื่อมต่ออินเตอร์เน็ตและไม่ต้องมีค่าใช้จ่ายใดๆ ง่ายต่อการติดตั้งและใช้งานโปรแกรม Xampp จะมาพร้อมกับ PHP ภาษาสำหรับพัฒนาเว็บแอปพลิเคชันที่เป็นที่นิยม MySQL ฐานข้อมูล, Apache จะทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ อีกทั้งยังมาพร้อมกับ OpenSSL , phpMyadmin (ระบบบริหารฐานข้อมูลที่พัฒนาโดย PHP เพื่อใช้เชื่อมต่อไปยังฐานข้อมูล) สนับสนุนฐานข้อมูล MySQL และ SQLite

https://www.apachefriends.org/index.html



Sublime Text 3 เป็นโปรแกรมเขียนโค้ดซึ่งสนับสนุนภาษาที่หลากหลาย เช่น C, C++, C#, CSS, D, Erlang, HTML, Groovy, Haskell, HTML, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, Matlab, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL, Textile และ XML

https://www.sublimetext.com/3

Composer เป็นเครื่องมือ ของ PHP ใช้จัดการ library ที่ต้องการใช้ในโปรเจ็ค ลักษณะการใช้งานคือ ให้เราระบุ library ที่โปรเจ็คของเราต้องการไว้ในไฟล์ composer.json จากนั้น composer จะทำการติดตั้งหรืออัพเดท library ที่เราต้องการให้เลย ช่วยให้เราจัดการกับ library ได้ง่ายขึ้น

https://getcomposer.org/

Laravel คือ PHP Framework รูปแบบ MVC และเป็นที่นิยมใช้มากของนักพัฒนาระบบ หรือเว็บแอพพลิเคชั่นในปัจจุบันเพราะมีความสามรถที่ช่วยในการทำงานให้ง่ายและเป็นระเบียบมากขึ้น

ข้อดี : Laravel เป็น framework ที่สามารถดึง class php สำเร็จรูปที่ดีที่มีคนเผยแพร่ในอินเตอร์เน็ตไว้ให้โหลดฟรีมาใช้งานได้จึงทำให้นักพัฒนาระบบหรือเว็บแอพพลิเคชั่นไม่ต้องเขียนโค้ดเองทั้งหมดโดยทำงานร่วมกับ composer ซึ่งเป็นโปรแกรมที่ให้เรามาติดตั้งบนเครื่องที่เราใช้งานประโยชน์ของมันคือช่วยให้เราดาวน์โหลดปลั๊กอินมาติดตั้งในโปรเจ็ค

ข้อเสีย : ถ้ามีการอัพเดทเวอร์ชั่นบางที่อาจทำให้มีบางไฟล์เปลี่ยนไปทำให้ไม่ตรงกับเวอร์ชั่นเดิม จึงอาจทำให้เกิดความวุ่นวายในการอัพเดทพอสมควร

Setup Laravel Project

```
composer create-project --prefer-dist laravel/laravel blog "5.4.*"
```

Run Laravel Project

```
php artisan serve
```

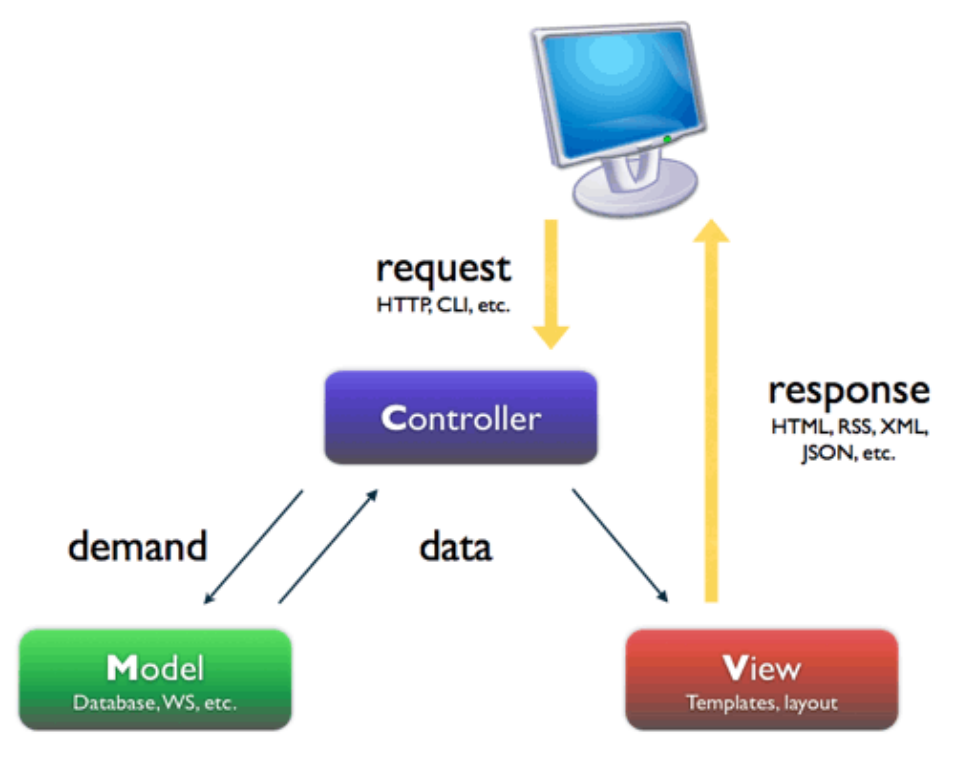
Laravel Framework

02.

Introduction to Laravel framework

ทำความรู้จักกับ Laravel Framework และ MVC Model

MVC Model



MVC เป็นสถาปัตยกรรมซอฟต์แวร์รูปแบบหนึ่ง ที่มีโครงสร้างแบ่งออกมาเป็น 3 ส่วนหลัก ตามตัวย่อของชื่อ ได้แก่ Model View และ Controller

1. Controller

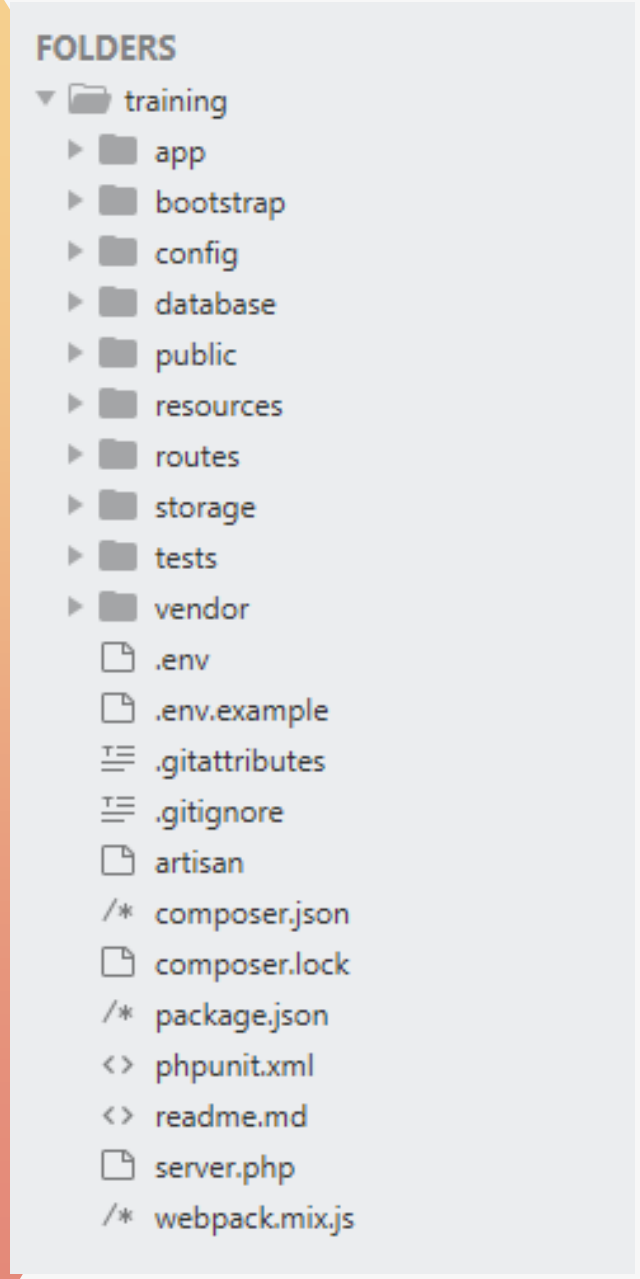
- เป็นส่วนที่ควบคุมการทำงานของโปรแกรม (Logic)
- เป็นส่วนที่ติดต่อการทำงานระหว่างผู้ใช้และโปรแกรม
- มีการติดต่อกับ Database(ฐานข้อมูล) ด้วย Model และแสดงผลข้อมูลผ่านทาง View
- เป็นส่วนที่มีการประมวลผลหลัก ของโปรแกรม

2. Model

- เป็นส่วนที่ติดต่อสื่อสารระหว่าง Object กับ Database (ฐานข้อมูล)
- เชื่อมความสัมพันธ์ของข้อมูล (Relationship)
- ตรวจสอบความถูกต้องของข้อมูล

3. View

- เป็นส่วนที่แสดงผลผ่าน Web browser ที่ให้ผู้ใช้งานมองเห็น
- โดยจะใช้ HTML แทรกด้วย script PHP
- การทำงานสัมพันธ์อยู่กับ Controller



Laravel Project Structure

app – เก็บส่วนหลักๆ ของโปรเจ็คอย่าง Controller และ Model ที่ใช้ในการประมวลผลและติดต่อกับฐานข้อมูล

config – เก็บส่วนการตั้งค่าต่างๆ ของโปรเจ็ค

database – เก็บส่วนที่จัดการกับ database ต่างๆ อย่าง migration และ seed data

public – เก็บส่วนที่ให้ user เห็นได้ เช่น ไฟล์ css, js และอื่นๆ และยังเก็บไฟล์ index.php ที่เป็นไฟล์เริ่มต้นของโปรเจ็ค

resources – เก็บส่วนของ View ทั้งหมด

routes – เก็บส่วนของไฟล์ที่ใช้กำหนดเส้นทางของ URL ที่ User เรียกเข้ามา

storage – เก็บส่วนของไฟล์ cache, session ต่างๆ

vendor – เก็บ Package ต่างๆ ที่ใช้ Composer ติดตั้งเข้ามา

.env – ไฟล์ที่ใช้ในการตั้งค่าส่วนต่างๆ ของโปรเจ็ค

Routing

การกำหนดเส้นทางของ URL และตรวจสอบสิทธิ์การเข้าถึงข้อมูล และกำหนดว่า Request ที่เข้ามานั้นจะทำงานต่ออย่างไร

```
Route::get(‘test’, function() {
        return “ทดสอบ”;
}
```

```
Route::post(‘test’, function() {
        return “ทดสอบ”;
}
```

```
Route::get(‘view’, function() {
        return view(‘welcome’);
}
```

```
Route::get(‘view’, function() {
        return view(‘welcome’);
}->name(‘view’);
```

```
Route::get('/exportExcel', 'FormController@exportExcel');
```

Routing Prefix

สามารถจัดกลุ่มของ Routing โดยอ้างอิงชื่อ Prefix โดยไม่ต้องสร้างชื่อเดิมซ้ำๆ

```
Route::prefix(‘product’)->group(function() {

        Route::get(‘get’, function() {
                return “Get Product”;
        }

        Route::post(‘create’, function() {
                return “Create Product”;
        }

});
```

```
Route::get(‘product/get’, function() {
        return “Get Product”;
}
```

```
Route::post(‘product/create’, function() {
        return “Create Product”;
}
```

Routing Parameter

ในการทำงาน หลายครั้งจะมีการส่งค่า parameter ต่างๆ มาทาง URL ด้วย เช่น www.moac.go.th/product/1234

```
Route::get('test/{id}', function($id) {
        return $id;
}
```

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

```
Route::get('test/{id}/{name}', function() {
        return $id;
}->where(['id' => '[0-9]+', 'name' => '[a-z]+']);
```

Controller

ส่วนของการจัดการ การทำงานต่างๆ ของระบบ (Logic) ซึ่งอยู่ระหว่างกลางของ Model และ View

การสร้าง Controller

```
php artisan make::controller TestController
```

controller ที่สร้างจะอยู่ที่โฟลเดอร์ App/Http/Controllers

```
public function test() {
        return "ทดสอบ";
}
public function testView() {
        return view("welcome");
}
public function testRedirect() {
        return redirect()->to('/')->with('status', 'success');
}
```

Routing Parameter To Controller

การส่งค่า Paremeter ต่างๆ ที่มาจาก URL ไปที่ Controller

```
Route::get(‘/test/{id}/{name}’, ‘TestController@test');
```

```
public function test($id, $name) {

        return $id . $name;
}

public function test(Request $request) {

        return $request->id . $request->name;
}
```

View

ส่วนที่ใช้ในการแสดงผลข้อมูลผ่านเว็บบราวเซอร์ ที่ User Request เข้ามาผ่านทาง URL

view จะอยู่ที่ resources/views

ทดลองสร้างไฟล์ php ในโฟลเดอร์ view โดยใช้นามสกุล .blade.php
จากนั้นสร้างเส้นทางสำหรับเรียก view ที่สร้างขึ้นมา

```
<html>
  <body>
    <h1> ทดสอบ View </h1>
  </body>
</html>
```

```
<html>
  <body>

   @if(…)
     ….
  @endif

  @foreach(…)
     …
  @endforeach

  </body>
</html>
```

Passing data to view

การส่งข้อมูลจาก Controller มาแสดงผลในส่วนของ view

```
Route::get(‘/view/{id}/{name}', ‘TestController@view');
```

```
public function view($Request $request) {
        $data = [‘id' => $request->id, ‘name' => $request->name];
        return view('test')->with($data);
}
```

```
<html>
  <body>
    <h1> ID = {{ $id }} </h1>
  <h1> Name = {{ $name }} </h1>
  </body>
</html>
```

Blade Layout

ในส่วนของ View จะมีการนำเอา Blade Template มาใช้งาน ซึ่งเป็น Template Engine ที่สามารถทำ Layout ต่างๆ แยกออกจากกัน แล้วนำมาประกอบกัน เพื่อสร้างเป็นหน้าเว็บเพจได้ โดยไม่ต้องเขียนโค้ดเหมือนเดิมซ้ำๆ

สร้างโฟลเดอร์ layout และสร้างไฟล์ master.blade.php เป็นไฟล์ Master ของ Layout

```
<html>
  <body>
            <h1> Header </h1>
            <h1> Content </h1>
             @yield('content')
            <h1> Footer </h1>
  </body>
</html>
```

@yield ใช้ในการสร้างตัวแปร สำหรับที่จะนำมาแสดงเนื้อหาใน Layout อื่นๆ

Blade Layout

สร้างไฟล์ display.blade.php เป็นไฟล์แสดงผลของ Layout

```
@extends('layout/master')

@section('content')

        <h1> แสดงส่วนของเนื้อหา </h1>

@endsection
```

@extends ใช้ในการเรียก Layout จาก master.blade.php มาใช้งาน

@section ใช้ในการแสดงผลส่วนของ @yield ที่สร้างไว้ใน master.blade.php โดยต้องทำการปิดด้วย @endsection

@include ใช้ในการดึงส่วนของ Layout นั้นๆ มาใช้งาน

Model

ส่วนของแบบจำลองแต่ละตารางในฐานข้อมูล ที่จะช่วยให้เราสามารถเชื่อมต่อและทำงานกับฐานข้อมูลได้ง่ายขึ้น

สร้างฐานข้อมูลใหม่โดยใช้ phpMyAdmin

```
Database name = training
Collation = utf8_general_ci
```

เชื่อมต่อกับฐานข้อมูล โดยตั้งค่าในไฟล์ .env

```
DB_CONNECTION=mysql
DB_HOST=127.0.01
DB_PORT=3306
DB_DATABASE=training
DB_USERNAME=root
DB_PASSWORD=
```

Migration

ส่วนของการจัดการตารางในฐานข้อมูล เช่น การเพิ่ม ลบ หรือแก้ไข ช่วยให้สามารถจัดการโครงสร้างในฐานข้อมูลได้อย่างเป็นระบบ

สร้างไฟล์ migration โดยใช้คำสั่ง

```
php artisan make:migration create_training_table
```

migrations ที่สร้างจะอยู่ที่โฟลเดอร์ database/migrations

```
public function up() {
        Schema::create('training', function (Blueprint $table) {
            $table->increments('id')->index();
            $table->string('name')->default("OHM");
            $table->string('surname', 100)->nullable();
            $table->integer('phone');
            $table->timestamp();
        });
}
```

```
public function down()
    {
        Schema::dropIfExists('training');
    }
```

Migration

เรียกใช้ฟังชัน up จากไฟล์ migration ด้วยคำสั่ง

```
php artisan migrate
```

เรียกใช้ฟังชัน down จากไฟล์ migration ด้วยคำสั่ง

```
php artisan migrate:rollback
```

Model Relationships

การสร้างความสัมพันธ์ของโมเดลแต่ละตารางในฐานข้อมูล ช่วยให้การเรียกใช้ข้อมูลจากฐานข้อมูลทำได้ง่ายยิ่งขึ้น

แก้ไขตาราง Training เพื่อเพิ่มความสัมพันธ์กับตาราง Users

```
php artisan make:migration update_training_table
```

```
public function up() {
        Schema::table ('training', function (Blueprint $table) {
            $table->integer(‘user_id’)->after(‘id’)->unsigned();
            $table->foreign('user_id')->references('id')->on(‘users');
        });
}
```

```
 public function down()
    {
         Schema::table('training', function (Blueprint $table) {
                    $table->dropForeign([‘user_id’]);
                    $table->dropColumn(‘user_id’);
         });
    }
```

Model Relationships

สร้าง Model ด้วยคำสั่ง

```
php artisan make:model Training
```

Model ที่สร้างจะอยู่ที่โฟลเดอร์ app

กำหนดชื่อตารางในฐานข้อมูลให้กับโมเดล

```
protected $table = "training";
```

Model Relationships

สร้างความสัมพันธ์ระหว่างตาราง Users และ Training

User Model

สร้างความสัมพันธ์แบบ one-to-one

```
public function training()
   {
      return $this->hasOne(Training::class, 'user_id');
   }
```

สร้างความสัมพันธ์แบบ one-to-many

```
public function training()
   {
      return $this->hasMany(Training::class, 'user_id');
   }
```

Model Relationships

สร้างความสัมพันธ์ระหว่างตาราง Users และ Training

Training Model

สร้างความสัมพันธ์กลับไปที่ User Model แบบ one-to-one หรือ one-to-many

```
public function user()
  {
     return $this->belongsTo(User::class);
  }
```

สร้างความสัมพันธ์กลับไปที่ User Model แบบ many-to-many

```
public function user()
  {
     return $this->belongsToMany(User::class);
  }
```

Eloquent

สำหรับ Model ใน Laravel จะมีการใช้ Eloquent ORM ที่จะช่วยให้การ Query ข้อมูลจากฐานมูลได้อย่างง่ายดาย และรวดเร็ว

สร้าง Routing ใหม่

```
Route::get('/database/get', ‘TestController@get');
Route::get('/database/insert', ‘TestController@insert');
Route::get('/database/update', ‘TestController@update');
Route::get('/database/delete', ‘TestController@delete');
```

การเรียกใช้งาน Model ใน Controller

```
use App\User;
use App\Training;
```

Eloquent

Insert Data

```
$user = new User;
$user->name = “OHM”;
$user->email = “nattapat_pr@opsmoac.go.th”;
$user->password = bcrypt(123456);
$user->save();
```

```
$training = new Training;
$training->user_id = $user->id;
$training->name = “Nattapat”;
$training->surname = “Pratchayatiwat”;
$training->save();
```

Eloquent

Get Data

```
$user = User::all();
```

```
$user = User::where('id', '=', 1)->get();
```

```
$user = User::find(1);
```

```
$user = User::with('training')->get();
```

```
$user = User::whereHas('training', function($q) {
        $q->where('id', 1);
})->get();
```

```
$user->name;
```

```
$user->training->name;
```

Eloquent

Update Data

```
$training = Training::where(‘id’, 1)->update([‘name’ => “N”, “surname” => “P”]);
```

```
$training = Training::find(1);
$training->name = “N”;
$training->surname = “P”;
$training->save();
```

Delete Data

```
Training::find(1)->delete();
```

```
$training = Training::find(1);
$training->delete();
```

Laravel Framework

03.

Workshop (Webboard Web Application)

การสร้างเว็บบอร์ดด้วย Laravel Framework

Workshop

Requirement

1. เว็บบอร์ดสามารถเขียนโพสต์ได้เฉพาะสมาชิกเท่านั้น
2. ผู้ที่ไม่ได้เป็นสามาชิกสามารถเข้าดูโพสต์ได้ แต่ไม่สามารถเขียนโพสต์ใหม่ และไม่สามารถแสดงความคิดเห็นได้
3. ทุกโพสต์ต้องผ่านการตรวจสอบจากผู้ดูแลระบบก่อน
4. เมื่อมีการแก้ไขโพสต์ต้องผ่านการตรวจสอบจากผู้ดูแลระบบอีกครั้ง
5. การลบโพสต์ สามารถทำได้โดยผู้แลระบบเท่านั้น
6. การเขียนโพสต์ และการเขียนแสดงความคิดเห็น ต้องสามารถปรับรูปแบบข้อความได้ แสดงรูปภาพได้

Workshop

Setup New Project

```
composer create-project --prefer-dist laravel/laravel blog "5.4.*"
```

Run Laravel Project

```
php artisan serve
```

Workshop

Database Design

สร้างตารางฐานข้อมูลโดยใช้ Migration จาก Diagram ด้านล่างนี้ จากนั้นสร้าง Model พร้อมทั้งกำหนดความสัมพันธ์

users	
id	int (PK)
name	varchar
email	varchar
password	varchar
remember_token	varchar
created_at	timestamp
update_at	timestamp

topic	
id	int (PK)
user_id	int (FK)
approve_status	char
subject	varchar
message	test
created_at	timestamp
updated_at	timestamp

comment	
id	int (PK)
topic_id	int (FK)
user_id	int (FK)
message	text
created_at	timestamp
updated_at	timestamp

Workshop

Authentication

สร้างระบบการตรวจสอบสิทธิ์การเข้าถึงข้อมูล ก่อนการเข้าใช้งานระบบ

สามารถสร้างระบบการตรวจสอบสิทธิ์ด้วยคำสั่ง

```
php artisan make:auth
```

Class Auth สามารถใช้ดึงข้อมูลของผู้ใช้งานที่อยู่ในระบบออกมาได้ทันที

```
use Auth;
Auth::check();
Auth::user()->id;
Auth::user()->name;
```

สร้าง Controller ใหม่

```
php artisan make:controller TopicController
```

Workshop

Authentication

เพิ่มการตรวจสอบสิทธิ์ของ controller ในส่วนของฟังก์ชัน construct

```
public function __construct() {
      $this->middleware('auth');
  }
```

เพิ่มฟังก์ชัน index เพื่อใช้ในการแสดงผลหน้าแรกของระบบ

```
public function index() {
     return view('home');
  }
```

เพิ่มการยกเว้นตรวจสอบสิทธิ์ให้กับฟังชันก์ index เพื่อให้ผู้ใช้งานทั่วไปเข้าถึงได้

```
public function __construct() {
     $this->middleware(‘auth’, [‘except’ => [‘index’]]);
  }
```

Workshop

Authentication

ปรับเส้นการแสดงผลหน้าแรกไปที่ TopicController@index

```
Route::get('/', 'TopicController@index');
```

Workshop

Config

ตั้งค่าระบบเบื้องต้น

ที่ไฟล์ .env

```
APP_NAME=Training-Workshop
APP_ENV=local
APP_DEBUG=true
APP_URL=http://127.0.0.1:8000
```

ที่ไฟล์ config/app.php

```
'timezone' => 'Asia/Bangkok',
```

Workshop

สร้างปุ่มสำหรับการเพิ่มโพสต์ใหม่ ที่ไฟล์ resources/view/home.blade.php

```
href="{{ url('post/create') }}"
```

สร้างเส้นให้กับการเพิ่มโพสต์ใหม่

```
Route::get(‘/post/create', ‘TopicController@create');
```

สร้างฟังก์ชัน create ใน TopicController

```
public function create()
   {
      return view('post/create');
   }
```

Workshop

สร้างโฟลเดอร์ post ที่ resources/view และสร้างไฟล์ create.blade.php ในโฟลเดอร์ post

สร้างฟอร์มสำหรับการโพสต์ใหม่ด้วย Blade Template จากนั้น Post ข้อมูลในฟอร์ม ไปที่ TopicController พร้อมทั้ง Insert ข้อมูลเข้าไปที่ Topic Model

Workshop

Form Validation

Controller สามารถตรวจสอบข้อมูลที่มาจากฟอร์มได้ว่าถูกต้องครบถ้วนหรือไม่

```
use Validator;


$this->validate($request, [
                'subject' => 'required|max:255|min:20’,
                ‘message' => 'required|min:100',
]);
```

Workshop

Install CKEditor Package

Package ที่ช่วยในการสร้าง Text Editor Form

```
composer require unisharp/laravel-ckeditor
```

ที่ไฟล์ config/app.php เพิ่ม package service ใน array ของ providers

```
Unisharp\Ckeditor\ServiceProvider::class,
```

Publish the resources เพื่อให้ง่ายต่อการจัดการกับ package

```
php artisan vendor:publish —tag=ckeditor —force
```

Workshop

Install CKEditor Package

ที่ไฟล์ Create Post ในโฟลเดอร์ resources/view/ post เพิ่ม ID ให้กับ Textarea

```
id="ckeditor"
```

เพิ่ม script ส่วนของ Footer

```
<script src="vendor/unisharp/laravel-ckeditor/ckeditor.js"></script>
```

```
<script> CKEDITOR.replace('ckeditor') </script>
```

Workshop

Install CKEditor Package

ที่ไฟล์ public/vendor/unisharp/config.js ปรับให้สามารถจัดตำแหน่งข้อความได้

```
config.extraPlugins = 'justify';
```

Workshop

Install File Manager Package

Package ที่ช่วยให้สามารถจัดกับการอัพโหลดไฟล์จากเครื่องได้ง่ายขึ้น สามารถใช้ร่วมกับ CKEditor ได้

```
composer require unisharp/laravel-filemanager
```

ที่ไฟล์ config/app.php เพิ่ม package service ใน array ของ providers

```
UniSharp\LaravelFilemanager\LaravelFilemanagerServiceProvider::class,
```

```
Intervention\Image\ImageServiceProvider::class,
```

เพิ่ม package service ใน array ของ aliases

```
'Image' => Intervention\Image\Facades\Image::class,
```

Workshop

Install File Manager Package

Publish the resources เพื่อให้ง่ายต่อการจัดการกับ package

```
php artisan vendor:publish --tag=lfm_config—force
```

```
php artisan vendor:publish --tag=lfm_public—force
```

```
php artisan vendor:publish --tag=lfm_view —force
```

Link โฟลเดอร์ storage เพื่อใช้ในการเก็บข้อมูลรูปภาพที่อัพโหลด

```
php artisan storage:link
```

Workshop

Install File Manager Package

สร้างเส้นทางสำหรับ Package File Manager ใน Routing

```
Route::group(['prefix' => 'laravel-filemanager', 'middleware' => ['web', 'auth']], function () {
    \UniSharp\LaravelFilemanager\Lfm::routes();
});
```

ที่ไฟล์ vendor/unisharp/laravel-filemanager/Lfm.php แก้ไข Path ของ Routing

```
\UniSharp\LaravelFilemanager\Controllers\
```

Workshop

Install File Manager Package

ที่ไฟล์ Create Post ในโฟลเดอร์ resources/view/ post เพิ่ม Script ส่วนของ Footer

```
var options = {
   filebrowserImageBrowseUrl: '/laravel-filemanager?type=Images',
   filebrowserImageUploadUrl: '/laravel-filemanager/upload?type=Images&_token=' + '{{ csrf_token() }}',
   filebrowserBrowseUrl: '/laravel-filemanager?type=Files',
   filebrowserUploadUrl: '/laravel-filemanager/upload?type=Files&_token=' + '{{ csrf_token() }}'
 };
```

ปรับส่วนของการเรียกฟังชันก์ CKEDITOR โดยเพิ่มการ options

```
CKEDITOR.replace( 'ckeditor', options);
```

Workshop

Middleware

สร้างระบบการตรวจสอบสิทธิ์การเข้าถึงข้อมูล ก่อนการเข้าใช้งานระบบ แบบหลายระดับชั้น

```
php artisan make:middleware AdminAuth
```

middleware ที่สร้างจะอยู่ที่โฟลเดอร์ app/Http/Middleware

```
use Auth;


public function handle($request, Closure $next)
   {
      if (Auth::user()->role == 1) {
         return $next($request);
      }else{
         return redirect('/');
      }
   }
```

Workshop

Middleware

แก้ไขตาราง Users เพิ่มคอลัมน์ role สำหรับใช้ในการกำหนดสิทธิ์ของแต่ละผู้ใช้งาน

```
php artisan make:migration update_users_table
```

```
public function up()
  {
     Schema::table ('users', function (Blueprint $table) {
        $table->char('role')->default('0')->after('id');
     });
  }
```

```
public function down()
  {
      Schema::table('users', function (Blueprint $table) {
         $table->dropColumn('role');
      });
  }
```

Workshop

Middleware

เพิ่มส่วนของ midleware ที่ได้สร้างใหม่ เข้าไปในการทำงานของระบบ

ที่ไฟล์ app/Http/Kernel.php เพิ่ม middleware ใน array ของ $routeMiddleware

```
'adminAuth' => \App\Http\Middleware\AdminAuth::class,
```

สร้างปุ่มสำหรับการเข้าไปที่หน้าอนุมัติโพสต์ของผู้ใช้งาน โดยปุ่มนี้ admin สามารถมองเห็นได้เท่านั้น

ที่ไฟล์ resources/view/home.blade.php

```
@if(Auth::user()->role == 1)


@endif
```

Workshop

Data Seeder

เพิ่มข้อมูลเข้าฐานข้อมูลโดยตรง ส่วนมากจะใช้เพื่อเพิ่มข้อมูลที่มีค่าคงที่ เช่น ข้อมูลจังหวัด และใช้กับการเพิ่มข้อมูล Dummy เพื่อทดสอบ

```
php artisan  make:seeder AdminSeeder
```

Seeder ที่สร้าง จะอยู่ในโฟลเดอร์ database/seeds

Insert Data ในฟังก์ชัน run

```
DB::table('users')->insert([
        'role' => 1,
        'name' => 'admin',
        'email' => 'admin@moac.go.th',
        'password' => bcrypt(123456),
]);
```

```
php artisan  db:seed --class=AdminSeeder
```

Workshop

Middleware

สร้างเส้นทางให้กับการอนุมัติโพสต์ของผู้ใช้งาน พร้อมทั้งกำหนดสิทธิ์การเข้าถึงเฉพาะ admin เท่านั้น

```
Route::group(['middleware' => ‘adminAuth'], function()
{
        Route::get(‘/admin/approve/list', ‘TopicController@approveList’);
}
```

```
Route::group([‘prefix’ => ‘admin’, 'middleware' => ‘adminAuth'], function()
{
        Route::get(‘/approve/list', ‘TopicController@approveList’);
});
```

Workshop

Middleware

สร้างฟังก์ชัน approveList ใน TopicController พร้อมทั้งดึงข้อมูลโพสต์ที่ยังไม่ได้อนุมัติจาก Model ไปแสดงที่ View

```
use App\Topic;
```

```
public function approveList() {
        $topic = Topic::where('approve_status', 'F')->get();
        return view('post/approve/list')->with('topic', $topic);
    }
```

สร้างโฟลเดอร์ approve ที่ resources/view/post และสร้างไฟล์ list.blade.php ในโฟลเดอร์ approve

```
@foreach($topic as $value) {
        {{ $topic->subject }}
        {{ $topic->user->name }}
}
```

Workshop

สร้างหน้า View สำหรับดูรายละเอียดของโพสต์ และสร้างปุ่มสำหรับการ Approve โพสต์จากนั้นทำการ POST ข้อมูลไปที่ Controller พร้อมทั้ง Update ข้อมูลใน Model

Workshop

แสดงข้อมูลของโพสต์ที่ได้รับการอนุมัติแล้วที่หน้า Home โดยดึงข้อมูลจาก Model และผู้ใช้งานทั่วไปสามารถมองเห็นได้

Workshop

Pagination

การแบ่งหน้าข้อมูลในกรณีที่ข้อมูลมีจำนวนมากๆ

```
public function approveList() {
        $topic = Topic::where('approve_status', 'T')->paginate(10);
        return view('/')->with($topic);
    }
```

```
@foreach($topic as $value) {
        $topic->subject;
        $topic->message;
}
```

```
{!! $topic->render() !!}
```

Workshop

สร้างหน้า View สำหรับดูรายละเอียดของโพสต์ โดยผู้ใช้งานทั่วไปสามารถมองห็นได้และสร้างฟอร์มสำหรับการแสดงความคิดเห็นสำหรับสามาชิกเท่านั้น จากนั้นให้ทำการ POST ข้อมูลความคิดเห็นไปที่ Controller เพื่อ Insert ข้อมูลเข้าสู่ Model พร้อมทั้งให้แสดงข้อมูลความคิดเห็นในหน้ารายละเอียดโพสต์นี้ด้วย

Workshop

สร้างหน้าโพสต์ของฉันและทำการดึงข้อมูลโพสต์ที่ผู้ใช้งานนั้นได้ทำการสร้างไว้มาแสดงผล พร้อมทั้งสร้างปุ่มสำหรับแก้ไขโพสต์ จากนั้นสร้างเส้นทางสำหรับการแก้ไขและโพสต์ไปที่ Controller เพื่อ Update ข้อมูลใน Model โดยต้องให้ผู้ดูแลระบบทำการตรวจสอบอีกครั้ง

Workshop

สร้างปุ่มสำหรับการ ลบโพสต์ ที่สามารถมองเห็นได้เฉพาะ admin เท่านั้น ในหน้า Home จากนั้นสร้างเส้นทางสำหรับการลบโพสต์ไปที่ Controller เพื่อ Delete ข้อมูลใน Model

Laravel Framework

04.

Deploy Laravel Project

การอัพโหลดโปรเจ็คขึ้นสู่เซิร์ฟเวอร์เพื่อใช้งานจริง

Deploy Laravel Project

Config

ตั้งค่าระบบก่อนการอัพโหลดขึ้นเซิร์ฟเวอร์

ที่ไฟล์ .env

```
APP_NAME=Training-Workshop
APP_ENV=production
APP_DEBUG=false
APP_URL=Domain…
```

Deploy Laravel Project

สร้าง PublicPathServiceProvider เพื่อให้ระบบสามารถค้นหาโฟลเดอร์ public ได้ หลังจากเปลี่ยน environment

```
php artisan make:provider PublicPathSerivceProvider
```

provider ที่สร้างจะอยู่ที่โฟลเดอร์ app/http/Providers แก้ไขในฟังก์ชัน register()

```
if ($this->app->environment() === 'local') {
        $this->app['path.public'] = public_path();
    } else if ($this->app->environment() === 'production') {
        $this->app['path.public'] = base_path().'/../'.env('PUBLIC_PATH');
    }
```

Deploy Laravel Project

เพิ่มส่วนของ provider ที่ได้สร้างใหม่ เข้าไปในการทำงานของระบบ

ที่ไฟล์ app/config.php เพิ่ม provider ใน array ของ providers

```
App\Providers\PublicPathServiceProvider::class,
```

Deploy Laravel Project

แก้ไขไฟล์ index.php เพื่อเปลี่ยน path การเข้าถึง root folder ของระบบ

ที่ไฟล์ public/index.php

```
require __DIR__.'/../../training/bootstrap/autoload.php';
```

```
function public_path($path = '')
{
    return realpath(__DIR__).($path ? DIRECTORY_SEPARATOR.$path : $path);
}
```

```
$app = require_once __DIR__.'/../../training/bootstrap/app.php';
```

Workshop

Upload Files

อัพโหลดโฟลเดอร์ public ไปไว้ที่โฟลเดอร์ htdocs

อัพโหลดโฟลเดอร์อื่นๆ ที่เหลือทั้งหมด ไปไว้ที่โฟลเดอร์ xampp (ระดับเดียวกับ htdocs)

Thank You