2SB07 PHP Framework Laravel Lab Report

240-203 Computer Engineering Software Lab II

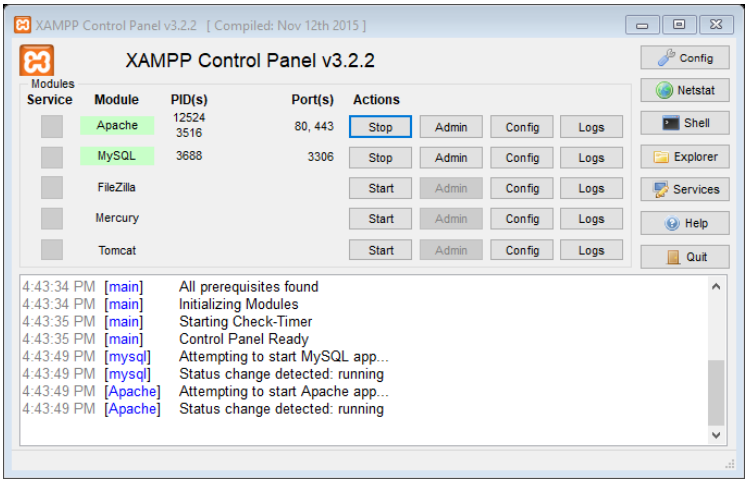
Student: ธีร์ธวัช สวาสดิ์ธรรม Student ID: 5910110150 Section: 01

การตั้งค่า Database

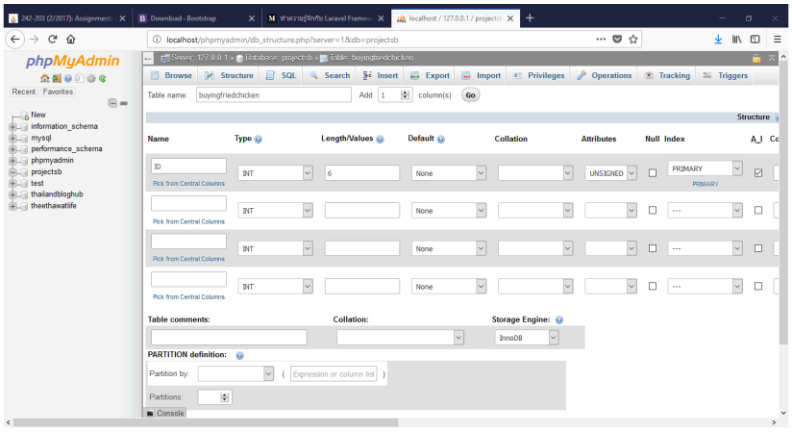
ในข้อนี้สามารถสร้าง Database ใหม่ก็ได้ หรือจะใช้ Database Guestbook เดิมแล้วสร้างตารางใหม่ หรือใช้ Database Guestbook เดิม ตารางเดิม และเปลี่ยนแปลงค่าก็ได้ ทางที่ดีควรจะใช้ Database เดิมก่อนเพื่อทดลองการทำงานของโปรแกรมเบื้องต้นของโปรแกรม

การทดลองแบบสร้าง Database ใหม่

1. เปิดใช้งาน XAMPP โดยกด Start ที่ Apache และ MySQL

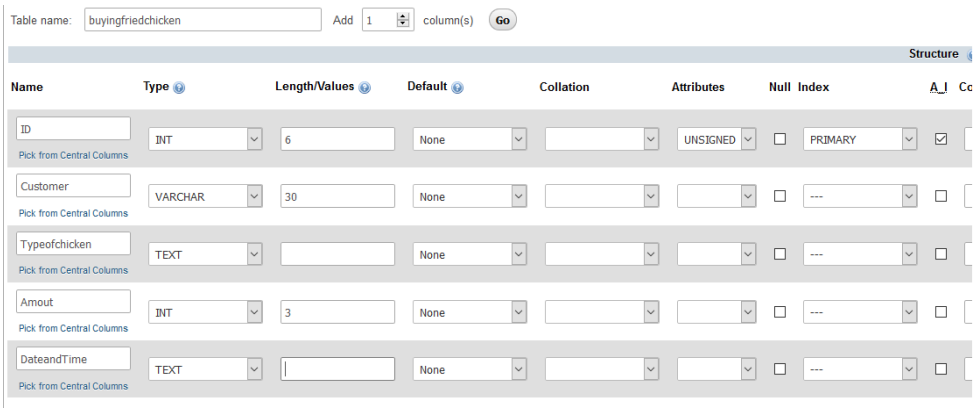


2. เปิดหน้าควบคุม Database โดยเข้าใน Web browser แล้วเข้าไปที่ localhost/phpMyAdmin หรือคลิกที่ Admin ในหน้า XAMPP Control Panel



3. สร้าง Database ขึ้นมาใหม่ โดยในที่นี้จะใช้ชื่อว่า guestbook จากนั้นสร้างตารางขึ้นมา ในที่นี้ให้ชื่อว่า buyingfriedchicken และกำหนดค่าต่างๆ ดังนี้

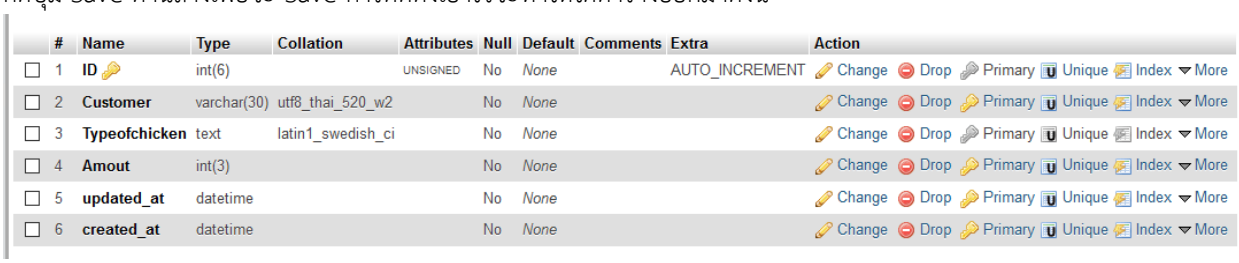
เนื่องจากโปรแกรมนี้คิดไว้ว่าจะเป็นโปรแกรมสำหรับร้านขายไก่ทอด จะกำหนดตัวแปล



ตารางแสดงข้อมูลส่วนต่างๆ ใน Table buyingfriedchicken

ชื่อ	ไว้สำหรับ	TYPE	ขนาด	อื่นๆ
ID	ใส่ ID ของข้อมูลต่างๆที่จะอินพุตเข้ามา	INT ตัวเลข	10 น่าจะหมายถึง 6 หลัก	ตั้งค่า Index เป็น Primary เพื่อที่จะล็อกไว้ไม่เท่ากับตัวอื่นๆ ตั้งค่า A_I หรือ Auto Increment เพื่อให้ระบบทำการเซทค่าเองแม้ว่าเราจะไม่ได้ใส่เอาไว้
CUSTOMER	ใส่ชื่อลูกค้า	VARCHAR ตัวอักษร	30 ไม่เกิน 30 ตัวอักษร	-
TYPEOFCHICKEN	ใส่ชนิดของไก่ทอด	TEXT ข้อความ	การใช้ TEXT ไม่จำเป็นต้องกำหนด	-
AMOUT	จำนวน	INT ตัวเลข	3	-
UPDATED_AT	ให้ระบบ stamp time ครั้งล่าสุดในการแก้ไข	datetime	ไม่จำเป็นต้องกำหนด	เหมือนจะเป็นข้อกำหนดของ Laravel เพราะมีการ Error เมื่อไม่ใส่ หรือเป็น Bug ที่เขียนติดอยู่ในส่วนหนึ่งของ Resource lab ของอาจารย์ที่ผมเอาออกไม่ได้
CREATED_AT	ให้ระบบ stamp time ครั้งแรกในการแก้ไข	datetime	ไม่จำเป็นต้องกำหนด	

4. กดปุ่ม Save ด้านล่างเพื่อจะ Save การติดตั้งเอาไว้จะทำให้ได้ตารางออกมาดังนี้



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ID	int(6)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	Customer	varchar(30)	utf8_thai_520_w2		No	None			Change Drop Primary Unique Index More
3	Typeofchicken	text	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	Amout	int(3)			No	None			Change Drop Primary Unique Index More
5	updated_at	datetime			No	None			Change Drop Primary Unique Index More
6	created_at	datetime			No	None			Change Drop Primary Unique Index More

หมายเหตุ เพื่อให้ Database รองรับภาษาไทยอาจจะต้องเปลี่ยน Collocation ในช่อง Customer เป็น utf_thai_520_w2

เพิ่มข้อมูลจากตาราง Comments ใน Resource lab โดยให้อยู่ในฐานข้อมูลเดียวกันกับตารางที่สร้างใหม่

1. เข้าไปที่ Database ที่สร้างขึ้นมาแล้วกด Import -> เรียกดู และไปหาตำแหน่งที่เก็บไฟล์ sql ของ Guestbook เอาไว้

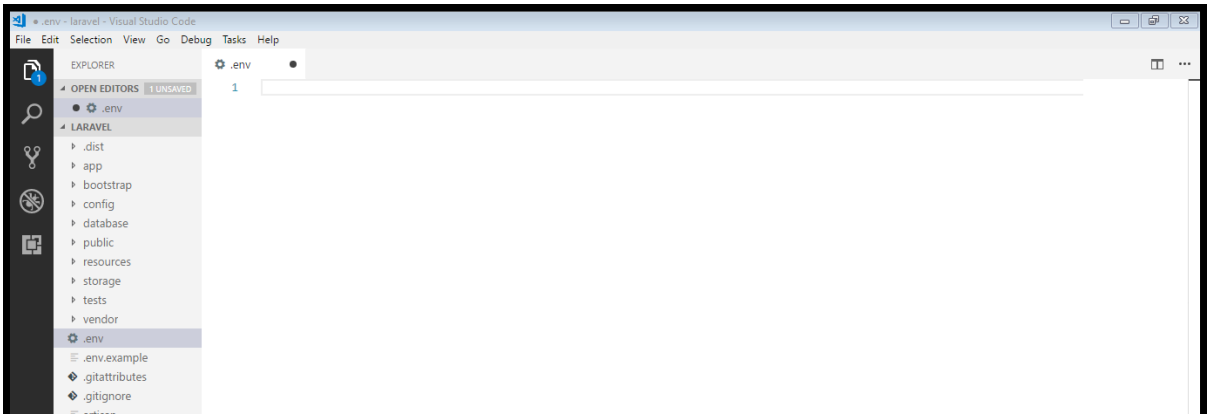
2. กดปุ่ม Go เพื่อทำการบันทึก

การ Config ค่าต่างๆ ใน Laravel

ในที่นี้จะใช้โปรแกรม Visual Studio Code (คนละตัวกับ Visual Studio 2017 หรือ อื่นๆ) เป็น Text-Editor เนื่องจากใช้ง่าย และสามารถดาวน์โหลดได้ฟรี ทั้งนี้สามารถใช้ text-editor อื่นๆ เช่น Sublime text, Notepad++, Editplus, AtomIDE หรือแม้กระทั่ง Visual Studio 2017 ตัวเต็ม หรือใช้ Notepad ก็ได้เช่นกัน

การเซทค่า .env

1. สร้างไฟล์ .env ขึ้นมา โดยสร้างใน text-editor แล้ว save เป็น .env



2. เขียนไฟล์ .env โดยอาจจะเอาโค้ดจากตัวอย่างที่ Laravel มีไว้ให้แล้วใน .env.example copy แล้วมาแก้ไข หรือนำโค้ดจาก Resource ของ Lab ก็จะคล้ายๆกันมีแค่ APP_KEY ที่แตกต่างกัน

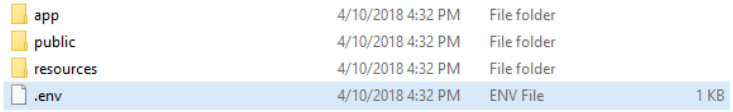
```
APP_ENV=local
APP_DEBUG=true
APP_KEY=ZawWBsBO07WDOKLtxHlT6K6CXtVXl7R3

DB_HOST=localhost
DB_DATABASE=guestbook
DB_USERNAME=root
DB_PASSWORD=

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

REDIS_HOST=localhost
REDIS_PASSWORD=null
REDIS_PORT=6379
```

โดยแก้ไขในช่วงขอ DB คือแก้ช่วง DB_DATABASE ให้เป็นชื่อดาต้าเบสของเรา แก้ DB_USERNAME ให้เป็น root และแก้ DB_PASSWORD ให้ว่างไว้ เนื่องจากใน Xampp Username เริ่มต้นจะเป็น root และ password จะไม่มี อย่างไรก็ดีถ้าหากจะเอาไปใช้ในเว็บไซต์จริงๆ ชื่อต่างๆก็จะต้องเป็นข้อมูลจริงตามระบบของเว็บไซต์ จากนั้นทำการบันทึกไว้ในโฟลเดอร์ Laravel

การเตรียมไฟล์จาก Guestbook มาใส่ใน Laravel

เพื่อให้ง่ายต่อการทำงานและศึกษา จะทำการเขียนโดยดัดแปลงจากระบบของ Guestbook ที่มีมาให้

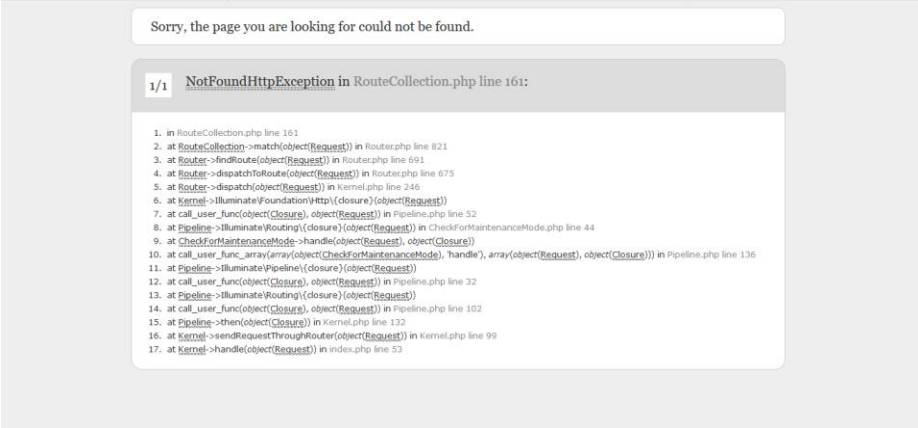
1. ในไฟล์ Guestbook ที่มีโค้ด app ที่เก็บ Controller และ Route อยู่ ใน Public มีส่วนที่เก็บ CSS และ Font จาก Bootstrap CSS Library อยู่ ซึ่งปัจจุบัน Bootstrap ได้ออกเวอร์ชั่นใหม่มาหลายตัว ถ้าต้องการเวอร์ชั่นใหม่ของ Bootstrap สามารถดาวน์โหลดได้ที่ getbootstrap.com และนำมาใช้แทนโฟลเดอร์ css ใน public ก็ได้



2. ทำการ Copy ไฟล์เหล่านี้เข้าไปในโฟลเดอร์ Laravel ที่อยู่ใน Xampp **ทดลองการ Run Website**

```
10  use App\Comment;
11  Route::get('myview', 'GuestbookController@reindex' );
12  Route::get('myview/index', 'GuestbookController@index' );
13  Route::post('myview/search', 'GuestbookController@search');
14
15  Route::post('myview/addComment','GuestbookController@addComment');
16  Route::get('myview/delete/{id}','GuestbookController@delete');
17  Route::get('myview/edit/{id}','GuestbookController@editComment');
18  Route::post('myview/saveComment/{id}','GuestbookController@saveComment');
```

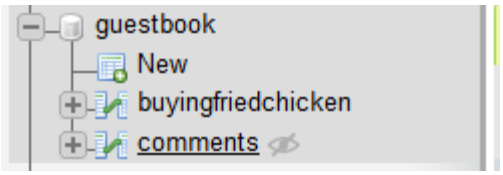
ในโค้ดได้ลิงค์หน้าของไฟล์ไว้ที่ /myview ดังนั้นเราก็จะไปหาไฟล์นี้ได้ที่ /laravel/public/myview โดยการทดลองนั้นได้เกิดปัญหาหน้า NotFoundHttpException หลายครั้ง จึงได้ค้นหาใน Google และได้คำตอบว่าอาจจะเป็นเพราะว่ารุ่นของLaravel และ Compiler อาจจะเก่าไป แก้ได้โดยการใช้คำสั่ง composer update และ php artisan clear-compiled โดยพิมพ์ลงใน Command Prompt ณ ตำแหน่งของโฟลเดอร์ที่เราจะแก้ หรือ โฟลเดอร์อะไรก็ได้ อย่างไรก็ดีต้องดาวน์โหลดและติดตั้งโปรแกรม Composer โดยดาวน์โหลดจาก https://getcomposer.org/



หลังจากแก้แล้ว ถ้ายังไม่ได้ให้ลบไฟล์เดิมแล้ว Copy ไฟล์ใน Laravel และ Guestbook เข้ามาใหม่

การปรับค่าใน Laravel เพื่อสร้าง Web Application ของตนเอง

1. ทำการเขียน Controller ใหม่ (Controller จะอยู่ที่ app/Http/Controllers) ชื่อ FriedchickenController และสร้าง template หน้า template.blade.php index.blade.php header.blade.php footer.blade.php ใหม่ อยู่ใน myview2 และเขียน Route ลิ้งค์ไปที่ FriedchickenController
    a. สร้างตารางของเราใน database guestbook โดยตารางจะชื่อว่า buyingfriedchicken (ทำในหน้า phpMyAdmin (localhost/phpmyadmin)) ซึ่งได้ทำแล้วในขั้นตอนข้างบน

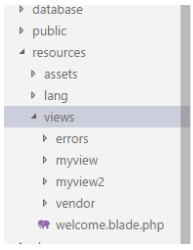


b. เข้าไปที่ ไฟล์ Route.php ใน app/Http ไปเพิ่มเติมส่วนที่ให้ลิงค์ไปที่ FriedchickenController โดยเริ่มที่ index และ reindex โดยล้อตาม lab resource ของอาจารย์

```
use App\Comment;
Route::get('myview', 'GuestbookController@reindex' );
Route::get('myview/index', 'GuestbookController@index' );
Route::post('myview/search', 'GuestbookController@search');

Route::post('myview/addComment','GuestbookController@addComment');
Route::get('myview/delete/{id}','GuestbookController@delete');
Route::get('myview/edit/{id}','GuestbookController@editComment');
Route::post('myview/saveComment/{id}','GuestbookController@saveComment');


Route::get('myview2','FriedchickenController@reindex');
Route::get('myview2/index','FriedchickenController@index');
```

c. สร้างไฟล์ FriedchickenController.php ใน app/Http/Controllers โดยสร้างล้อจาก Guestbook Controller โดยสร้างแค่ฟังก์ชัน reindex ก่อน

2. เตรียม resource สำหรับหน้าเว็บไซต์ สร้างโฟลเดอร์ myview2 คู่กับ myview ใน resources/views/



a. เข้าไปในโฟลเดอร์และสร้างไฟล์ templete.blade.php index.blade.php footer.blade.php header.blade.php เป็นพื้นฐานไว้ก่อน โดย templete.blase.php ใส่ค่าดังนี้

```
<!DOCTYPE html>
<html lang="en">
<head>
//ส่วนใน head นี้ไม่จำเป็น ส่วนที่จำเป็นอยู่ใน body
<title>Document</title>
</head>
<body>
    @include("myview2.header")

    @yield("content")

    @include("myview2.footer")

</body>
</html>
```

ถ้าใช้ Visual Studio Code สามารถเรียกแทกนี้อัตโนมัติได้โดยพิมพ์ ! แล้วกด Enter

b. สร้างหน้า index.blade.php ซึ่งเป็นส่วนหลักของหน้าเว็บไซต์โดยใส่

```
@extends(“myview2.template”)

@section(“ใส่ชื่อ Section”) // ใส่เป็น Content

…

@endsection
```

c. สร้างหน้า header.blade.php และ footer.blade.php ซึ่งจะเป็นส่วนหัวและส่วนล่างของหน้าเว็บไซต์โดยมีลักษณะเป็น

```
@section(“ใส่ชื่อ Section”) // ใส่เป็น header หรือ footer

…

@show
```

3. เริ่มออกแบบเว็บไซต์

a. เตรียมหน้าเว็บ โดยดึงเทมเพลตจาก Bootstrap เข้ามา โดยการเขียน tag html ในส่วน head ของหน้า templete.blade.php โดยใน Resource จะมีเทมเพลทของ Bootstrap ให้แล้วแต่อาจจะเป็นเวอร์ชั่นเก่า โดยสามารถใส่เข้าแล้วใช้ได้เลย โดย templete css จะเก็บไว้ที่หน้า public/css

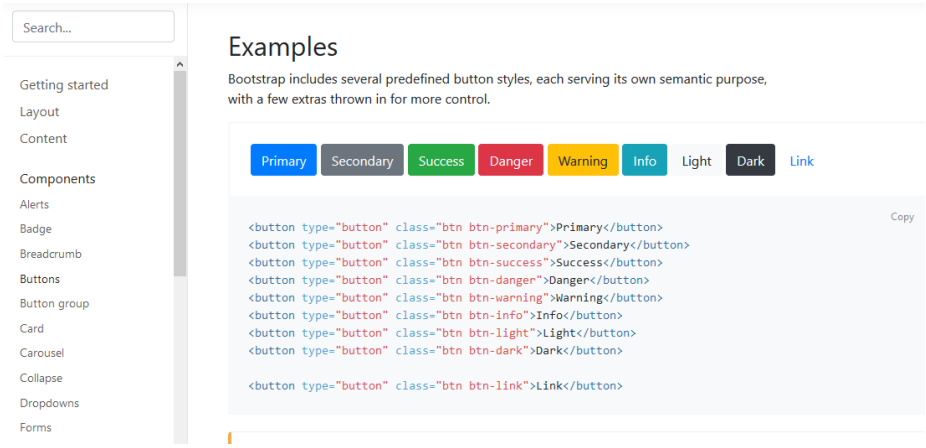
```
<link rel="stylesheet" href="{{URL::asset('/css/bootstrap.min.css')}}">
```

ซึ่งเนื่องจากเวอร์ชั่นของ Bootstrap CSS ใน Resource ค่อนข้างจะเก่าแล้ว จึงทำการดาวน์โหลดเวอร์ชั่นล่าสุดมาจากเว็บไซต์ของ Bootstrap ที่ https://getbootstrap.com

จากนั้นแตกไฟล์ออกมา และสามารถนำมาใส่แทนไฟล์ CSS ของ bootstrap รุ่นเก่าเลย แต่ในที่นี้จะใส่ใน โฟลเดอร์ Bootstrap 4.1 และนำตำแหน่งมาเขียนแท็ก CSS ได้ดังนี้

```
<head>
    <!--ใน Head ส่วนนี้ไม่จำเป็นต้องเขียน -->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="{{URL::asset('/Bootstrap 4.1/css/bootstrap.min.css')}}">
    <link rel="stylesheet" href="{{URL::asset('/Bootstrap 4.1/css/bootstrap.css')}}">
    <link rel="stylesheet" href="{{URL::asset('/Bootstrap 4.1/css/bootstrap-grid.css')}}">
    <link rel="stylesheet" href="{{URL::asset('/Bootstrap 4.1/css/bootstrap-grid.min.css')}}">

</head>
```

b. ดูรูปแบบการจัดวางต่างๆ ได้ จาก https://getbootstrap.com/docs/4.1/getting-started/introduction/ ทั้งที่ Component และ Layout



c. นำมาออกแบบโครงสร้างหลักๆ ของหน้าต่างๆ

4. ดึงข้อมูลเข้าสู่เว็บไซต์

a. เริ่มเขียน Route ลิงค์ไปที่ฟังก์ชัน index ใน Controller

```
20
21    Route::get('myview2','FriedchickenController@reindex');
22    Route::get('myview2/index','FriedchickenController@index');
```

b. เริ่มเขียน Controller (Controller จะอยู่ที่ app/Http/Controllers) เพิ่มในส่วนของตัว use เพราะเราต้องใช้ฟังก์ชันคำสั่งเหล่านี้

```
2
3   namespace App\Http\Controllers;
4
5   use App\Comment; //ลิงค์คำมาจากไฟล์คอมเมนต์
6   use Illuminate\Support\Facades\Input; //ใช้คำสั่ง Input
7   use Illuminate\Support\Facades\DB;  //ใช้คำสั่ง DB หรือ Database
8   use Illuminate\Support\Facades\Redirect; //ใช้คำสั่ง Redirect
9
```

c. เขียนในส่วนของ Index โดยให้นำข้อมูลจาก database เข้ามาเก็บไว้ในตัวแปรเพื่อจะเอาไว้ใช้ในการเขียนเว็บไซต์ต่อไป โดยให้ตัวแปร $friedchicken รับค่าข้อมูลจากการวิเคราะห์ของไฟล์ Comment ที่อยู่ที่

App/comment

```
class FriedchickenController extends Controller{
    public function index(){
        $friedchicken=Comment::orderBy('ID','DESC')->get();//ดึงค่าจาก Comment เรียงตาม ID แบบ DESC
        return view('myview2.index') //ส่งค่าออกเป็นหน้า myview2.index
            ->with('title','Guestbook buyingfriedchicken')
            ->with('buyingfriedchicken',$friedchicken); //ส่งค่าตัวแปร friedchicken ออกไปในชื่อ buyingfriedchicken
    }
```

d. กำหนดค่า Comment ให้ลิงค์ไปที่ตารางของเรา Comment ลิงค์ไปที่ดาต้าเบสของเราแน่นอนอยู่แล้ว แต่ยังไม่ได้ลิงค์ไปที่ตารางของเรา ดังนั้นเราต้องไปกำหนดค่าในไฟล์ Comment.php ซึ่งอยู่ในโฟลเดอร์ App โดยใช้คำสั่ง protected จากนั้นทุกครั้งที่เราใช้คำสั่ง Comment ก็จะเหมือนกับเรียก table นี้ ซึ่ง Table ในครั้งนี้ชื่อว่า buyingfriedchicken

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Comment extends Model
{
    protected $table = 'buyingfriedchicken';
}

```

5. เขียนข้อมูลในหน้าแรก
   a. ไปที่ index.blade.php ทำการเขียนโค้ดลงไป โดยเขียนลงไประหว่างส่วน section กับ endsection
   b. ตั้ง @foreach ...@endforeach ซึ่งจะเป็น loop จะสามารถเอามาแสดงผลข้อมูลที่มาจาก database ได้ภายใน loop นี้ และที่ @foreach เราสามารถกำหนดได้ว่า ให้ตัวแปร $buyingfriedchicken ที่เราส่งมาจาก Controller แทนด้วย $buy โดยการเรียกค่าจากตาราง หรือ ตัวแปรที่เราส่งมาโดยใช้คำสั่ง {{$buy->.....}} โดยที่ชี้ไปยังชื่อ column ในตาราง

```
@extends("myview2.template")
@section("content")
<div class="container">
<div class="row">
<div class="col-sm-8">
<h5>รายการไก่ที่สั่ง</h5>
@foreach($buyingfriedchicken as $buy)
<div class="card">
<div class="card-body">
    <div class="flex" style="display:flex;">
    <div style="width:90%;" >
      รหัสรายการ {{$buy->ID}} ชื่อผู้สั่ง {{$buy->Customer}}
      <h4>รูปแบบไก่ {{$buy->Typeofchicken}}  จำนวน {{$buy->Amout}} ชิ้น </h4>
      <p>เวลาที่สั่งซื้อ {{$buy->updated_at}} </p>
      </div>
      <a href="editOrder/{{$buy->ID}}"><button class="btn btn-light">แก้ไข</button></a>
      <a href="delete/{{$buy->ID}}"><button class="btn btn-light">ลบ</button></a>
      </div>
  </div> </div>
<br>

@endforeach
```

   c. หลังจากจบ foreach จะเป็นส่วนอื่นๆ ของหน้า ซึ่งเราสามารถตกแต่งได้โดยใช้ HTML และ CSS พื้นฐาน โดยสามารถเอารูปแบบ Grid ของเว็บไซต์(การแบ่งหน้าเว็บเป็นส่วนๆ คอลลัมม์ต่างๆ) มาได้จาก Bootstrap และ

ที่สำคัญคือการทำแบบฟอร์มรับข้อมูล

```
<div class="col-sm-4">
<h5>สั่งซื้อไก่ทอด</h5>
<p>กรอกข้อมูลตามแบบฟอร์มต่อไปนี้</p>
<div class="card"><div class="card-body">


<form role="form"  action="addorder" method="POST">
ชื่อผู้สั่ง
<input type="text" class="form-control" required name="customer" id="customer">
รูปแบบไก่

  <select class="form-control" name="typeofchicken" id="typeofchicken">
    <option value="Original">Original รสดั้งเดิม </option>
    <option value="Crispy">Crispy กรอบ </option>
    <option value="Black Peper">Black Peper พริกไทยดำ</option>
    <option value="Fish Source"> Fish Source ไก่ทอดน้ำปลา</option>
  </select>

จำนวนที่จะสั่งซื้อ
<div style="display:flex;">
<input type="number" class="form-control" style="width:30%;" name="amout" id="amout">
ชิ้น
</div>
<br>
<button type="submit" class="btn btn-primary">สั่งซื้อ</button>

</form>
```

d. จะเห็นได้ว่าได้ส่ง action ของ form ไปที่ addorder ซึ่งจะทำในข้อต่อไป

6. เพิ่มข้อมูลเข้าไปใน Database โดยลิงค์จาก from และส่งไปที่ addorder โดยข้อมูลที่จะส่งมีสามตัวคือ customer typeofchicken และ amout

a. เขียน route สำหรับกำหนดเส้นทานไปของ addorder ให้ไปที่ FriedchickenController

```
Route::post('myview2/addorder','FriedchickenController@addorder');
```

b. เขียน function addorder ใน FriedchickenController ตามรูปแบบดังนี้

```
public function addorder(){
    $friedchicken= new Comment;
    //รับค่าข้อมูล
    $friedchicken->Customer=Input::get('customer');
    $friedchicken->Typeofchicken=Input::get('typeofchicken');
    $friedchicken->Amout=Input::get('amout');
    //บันทึกค่าข้อมูล
    $friedchicken->save();
    //Redirect กลับไปหน้าเดิม
    return Redirect::to('myview2/index');
}
```

7. การทำฟังก์ชั่น delete จากข้อ 5b จะเห็นได้ว่าเราได้สร้างปุ่มลบ และลิงค์ไปที่ delete/{{$buy->ID}} โดยใช้คำสั่ง a href ทั่วไป แต่เรานำ ID มาด้วยซึ่งจะเอามาใช้วิเคราะห์อีกที

a. ทำการกำหนด Route ของการ delete ให้ลิงค์ไปที่ Controller

```
Route::get('myview2/delete/{id}','FriedchickenController@delete');
```

b. เขียนฟังก์ชันควบคุมการลบใน controller โดยฟังก์ชันมีการรับค่าของ id ไว้ด้วย โดยให้ Comment ทำการ Search หา item ที่มี id นี้

```
public function delete($id){
    $friedchicken=Comment::find($id);
    DB::table('buyingfriedchicken')->where('ID', '=', $id)->delete();
   return Redirect::to('myview2/index');
}
```

8. การทำฟังก์ชัน Edit และ ฟังก์ชัน Save ค่าของการ edit จากข้อ 5b จะเห็นได้ว่าเราได้สร้างปุ่มedit และลิงค์ไปที่ editOrder/{{$buy->ID}}

   a. กำหนด Route สำหรับ editOrder

```
25  Route::get('myview2/editOrder/{id}','FriedchickenController@editOrder');
```

   b. สร้าง Controller สำหรับ editOrder โดยเริ่มจากหาค่าจาก ID มาเก็บไว้ที่ตัวแปรก่อน แล้วส่งค่าไปที่ edit.blade.php หรือ myview2.edit คือใช้โครงสร้าง myview2 แต่แทนที่จะเป็น index จะเป็น edit แทน

```
38      public function editOrder($id){
39          $friedchicken=Comment::find($id);
40          return view('myview2.edit')
41          ->with('buyingfriedchicken',$friedchicken);
42      }
```

   c. ดีไซน์หน้าเว็บ edit.blade.html โดยจัดเป็น section content เช่นเดียวกับ index สร้างเป็นฟอร์ม และตั้งค่าที่ได้จาก controller (ค่าใน Database) เป็นค่า default ของ form เนื่องจากเราไม่ได้ใช้ foreach แล้วดังนั้น ก็จะต้องใช้ชื่อเต็มที่แทนตัวแปร $friedchicken ของ Controller คือ $buyingfriedchicken โดยฟอร์มจะส่งไปที่ submitted/($buyingfriedchicken->ID)

```
1   @extends("myview2.template")
2   @section("content")
3   <div class="container">
4   <div class="col-sm-8">
5   <form role="form"  action="../submitEdit/{{$buyingfriedchicken->ID}}" method="POST">
6   ชื่อผู้สั่ง
7   <input type="text" class="form-control" required name="customer" id="customer" value="{{$buyingfriedchicken->Customer}}">
8   รูปแบบไก่
9
10    <select class="form-control" name="typeofchicken" id="typeofchicken" value="{{$buyingfriedchicken->typeofchicken}}">
11      <option value="Original">Original รสดั้งเดิม </option>
12      <option value="Crispy">Crispy กรอบ </option>
13      <option value="Black Peper">Black Peper พริกไทยดำ</option>
14      <option value="Fish Source"> Fish Source ไก่ทอดน้ำปลา</option>
15    </select>
16
17  จำนวนที่จะสั่งซื้อ
18  <div style="display:flex;">
19  <input type="number" class="form-control" style="width:30%;" name="amout" id="amout" value="{{$buyingfriedchicken->Amout}}"
20  ชิ้น
21  </div>
22  <br>
23  <button type="submit" class="btn btn-primary">แก้ไขการสั่งซื้อ</button>
24
25  </form>
26  </div>
27  </div>
28  <br>
29  @endsection
```

   d. เขียน Route ของ submitted ซึ่งมีการส่งค่าของ ID มาด้วย

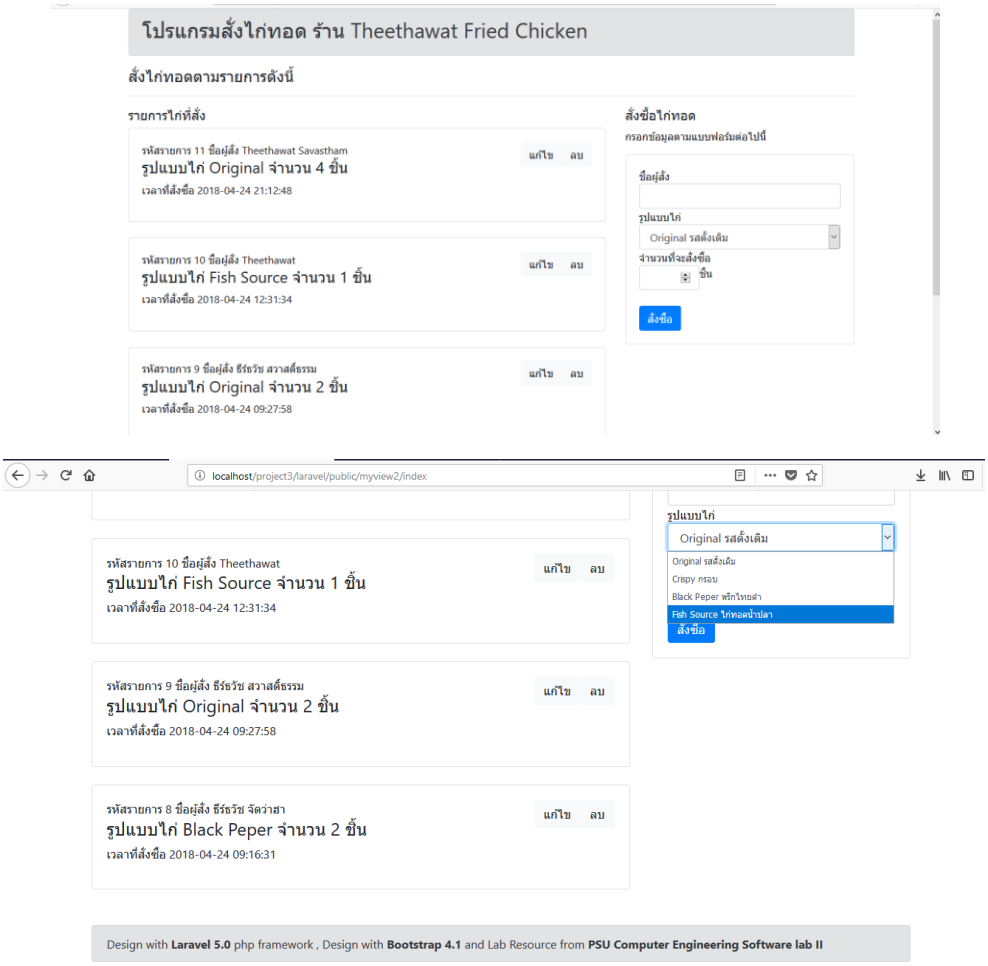
```
26  Route::post('myview2/submitEdit/{id}','FriedchickenController@submitEdit');
```

   e. เขียน Controller ของ submitted เพื่อทำการอัพเดทข้อมูลในตาราง

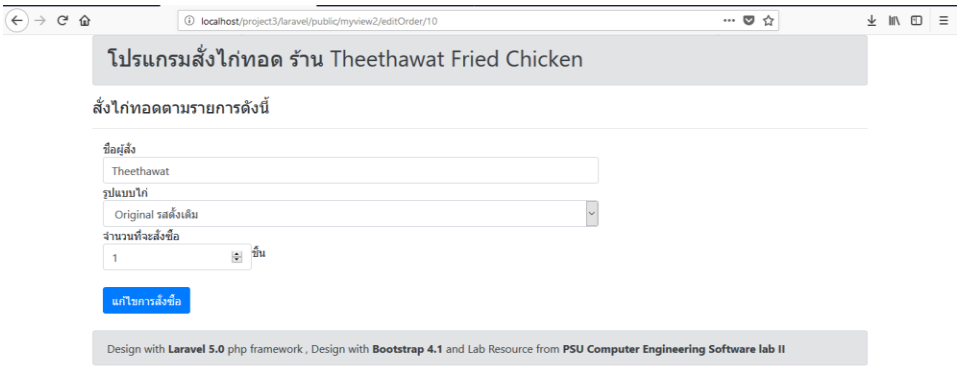
```
44      public function submitEdit($id){
45          $friedchicken=Comment::find($id);
46          $friedchicken->Customer=Input::get('customer');
47          $friedchicken->Typeofchicken=Input::get('typeofchicken');
48          $friedchicken->Amout=Input::get('amout');
49          $friedchicken->save();
50          DB::table('buyingfriedchicken')
51              ->where('ID', $id)
52              ->update(['Customer' => $friedchicken->Customer,'Typeofchicken' => $friedchicken->Typeofchicken,
53              'Amout' => $friedchicken->Amout]);
54          return Redirect::to('myview2/index');
55      }
```

รูปแบบเว็บไซต์ที่ออกมา (การจะเข้าไปที่เว็บไซต์หน้านี้ต้องไปที่ /myview2)

- หน้า index



- หน้าแก้ไข



แหล่งข้อมูลอ้างอิง และ Resource

- LAB Sheet 2SB07 and Guestbook Resource Instructor Seksun Suwanmanee /Warodom Werapun
- https://laravel.io/forum/04-08-2015-adding-data-to-database-using-eloquent-model

- https://laravel.com/docs/5.6/
- https://getbootstrap.com/docs/4.1/
- https://laracasts.com/discuss/channels/general-discussion/notfoundhttpexception-on-every-route-1?page=1
- https://getcomposer.org/

หมายเหตุ รายงานฉบับนี้ไม่ได้ลบ Hyperlink ออก อาจจะเป็นการจัดรูปแบบที่ไม่ดีนัก แต่เนื่องจากเป็นไฟล์ pdf จะทำให้สามารถกดลิงค์ต่อไปได้