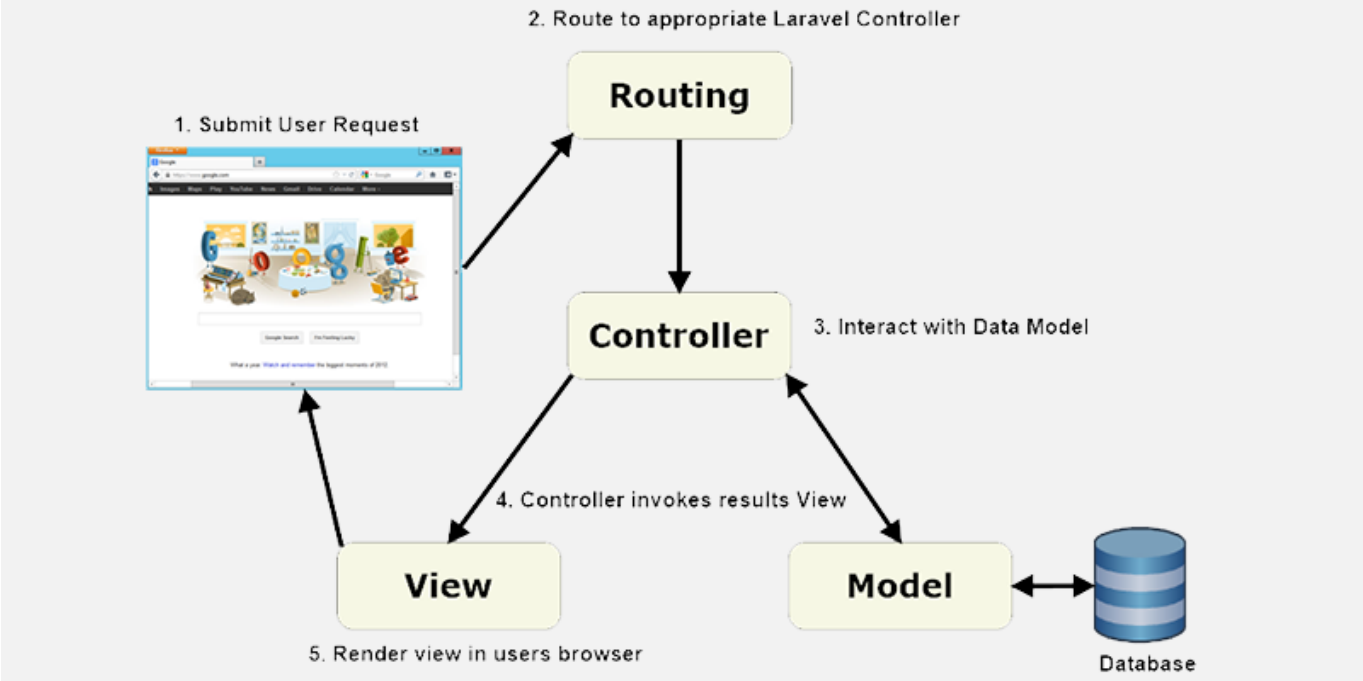
BUỔI 03: **CONTROLLER TRONG LARAVEL**

5.1/ Mô hình MVC trong Laravel



Mọi Request từ phía người dùng đều phải qua Route, dữ liệu được gửi xuống Controller để xử lý, cần dữ liệu sẽ lấy từ Model lên hoặc cập nhật dữ liệu xuống Model, kết quả gửi ra View cho người sử dụng.

MVC là viết tắt của ba từ Model – View – Controller. Trong đó:

- Model: cấu trúc dữ liệu theo cách tin cậy và chuẩn bị dữ liệu theo lệnh của controller
- View: Hiển thị dữ liệu cho người dùng theo cách dễ hiểu dựa trên hành động của người dùng.
- Controller Nhận lệnh từ người dùng, gửi lệnh đến cho Model để cập nhập dữ liệu, truyền lệnh đến View để cập nhập giao diện hiển thị.

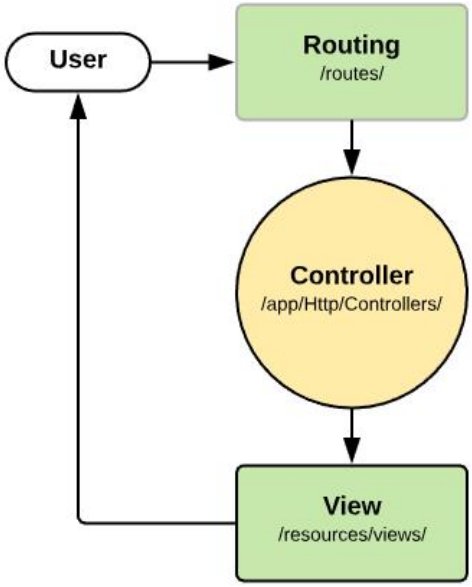
5.2/ Controller trong Laravel

Đặt vấn đề: Khi cần xử lý hàng loạt thao tác như: tính toán giá trị, xử lý lưu trữ dữ liệu, ... thì không thể nào chỉ làm bên trong **FUNCTION** của một Routing được, việc xử lý sẽ được thực hiện bên trong **FUNCTION (Method)** của một **Controller** và trả kết quả về View để hiển thị ra trình duyệt.

5.2.1/ Controller là gì ?

- Để xử lý logic (tính toán) thì chúng ta có thể viết trong nhiều chỗ khác nhau, nhưng tốt nhất là *gom chung viết bên trong Controller*, đặc biệt là các tính toán dài, phức tạp.
- Controller sinh ra là để trở thành trung tâm xử lý logic, nên cần phải tận dụng, thay vì viết tính toán ở nơi khác.
- Controller được đặt bên trong thư mục /app/Http/Controllers/.

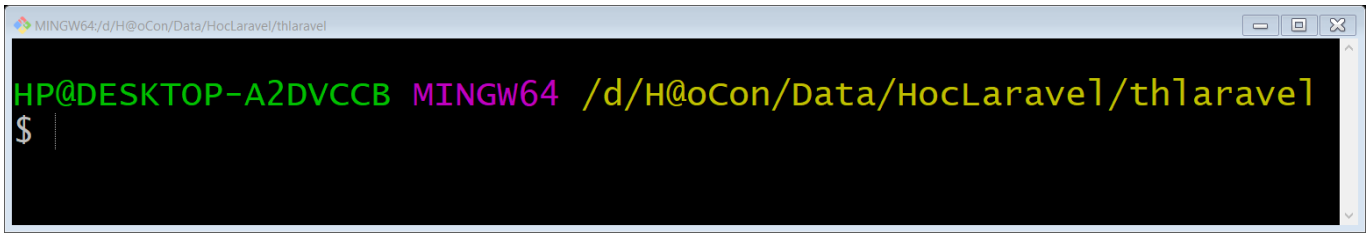
Để hiểu một cách cơ bản nhất cách hoạt động của một Controller ta xem cấu trúc đơn giản sau:



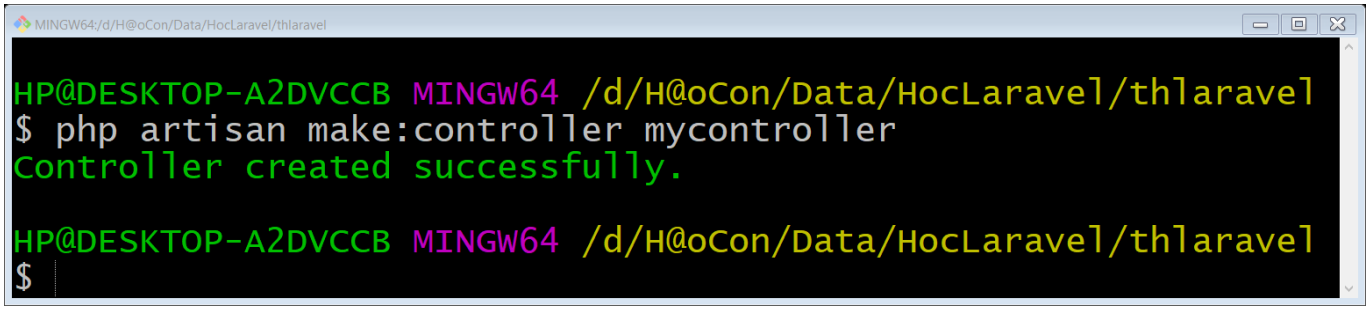
Folder / File	Mô tả
User	User gửi yêu cầu.
Routing	Yêu cầu gửi từ User sẽ được Routing điều hướng: Tới Controller để xử lý yêu cầu. Tới thẳng View nếu không cần xử lý.
Controller	Controller được xem như trung tâm điều khiển của hệ thống, tất cả thao tác xử lý nên được thực hiện ở đây. Kết quả xử lý sẽ được trả về view.
View	View nhận dữ liệu xử lý từ Controller (hoặc Routing), hiển thị kết quả cho người dùng.

5.2.2/ TẠO MỘT CONTROLLER

Bước 01: Trước tiên di chuyển tới thư mục myproject và mở cửa sổ lệnh cmd:



Bước 02: Tạo Controller bằng lệnh Artisan: php artisan make:controller mycontroller



- php artisan - Công cụ hỗ trợ viết command line tích hợp sẵn trong Laravel, sẽ còn gặp lại nhiều.
- make:controller - Lệnh tạo Controller.
- mycontroller - Tên Controller do mình tự đặt.

Nếu xuất hiện thông báo "Controller created successfully." thì quá trình tạo file thành công. Lúc này xem trong thư mục /app/Http/Controllers/ thì ta sẽ thấy tồn tại file MyController.php với nội dung bên trong mặc định như sau:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class mycontroller extends Controller
{
    //
}
```

namespace App\Http\Controllers;	Là tên dùng để phân biệt những Controller có tên giống nhau, tuy nhiên thay vì dùng tên đại diện riêng, thì lại dùng đường dẫn thư mục tới file Controller App\Http\Controllers, do đó với namespace ta cần ghi đầy đủ cấu trúc thư mục chứa file Controller là được.
use Illuminate\Http\Request;	Khi khai báo, chúng ta có thể sử dụng được các Request (như Get, Post, ...), hoặc dùng để gọi, quản lý các Class.
mycontroller	là tên class, tên này thường trùng với tên file Controller vừa tạo. Nội dung Controller được viết bên trong dấu ngoặc móc {}

Ghi chú: Ngoài cách tạo file Controller bên trên, còn có thể tạo file đơn giản hơn bằng cách copy file Controller đã có sẵn hoặc tạo file PHP đặt bên trong thư mục App\Http\Controllers, sau đó viết nội dung như bên trên là được.

5.2.3/ CHẠY MỘT CONTROLLER

Ví dụ hiển thị một trang Views từ việc điều khiển phần Routing gửi yêu cầu tới Controller xử lý, xong kết quả sẽ trả lại về Views. Viết một phương thức **chaoban** trong file **mycontroller.php** vừa tạo với nội dung sau:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class mycontroller extends Controller
{
    public function chaoban()
    {
        return view("Buoi05.chao");
    }
}
```

Tạo thêm một file View **chao.blade.php** đặt bên trong thư mục **/resources/views/Buoi05**, với nội dung:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
        crossorigin="anonymous">
    <title>Chào Bạn</title>
</head>
<body>
    <h1 class="alert alert-primary">LARAVEL chào mừng các bạn đến với Website này</h1>

<!-- Option 1: Bootstrap Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
    crossorigin="anonymous"></script>
</body>
</html>
```

Viết một routing /routes/web.php đơn giản hiển thị trang

- **Trường hợp 01: sử dụng Laravel 07 trở về trước**

```
Route::get('/chao', 'mycontroller@chaoban');
```

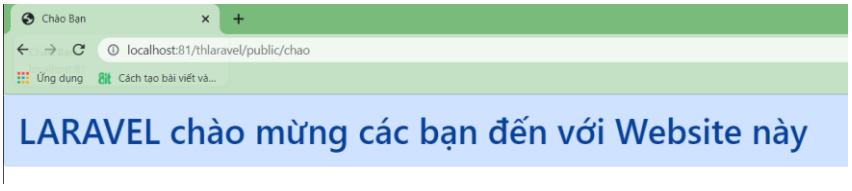
Phân tích:

- '/chao': là đường dẫn trên trình duyệt.
- 'mycontroller@chao' : mycontroller là tên controller và chao là tên function của Controller.

- **Trường hợp 02: sử dụng Laravel 08 trở đi**

```
Route::get('/chao', [mycontroller::class, "chaoban"]);
```

Hiển thị trình duyệt



Chúng ta thấy nội dung trang chao.blade.php đã được gọi thông qua mycontroller.php. Khi gõ lên thanh địa chỉ trình duyệt nội dung http://localhost/public/chao, khi này Routing sẽ nhận yêu cầu xử lý, Routing chuyển nội dung xử lý sang mycontroller với function chaoban, sau khi Controller xử lý xong sẽ trả kết quả về View

5.2.4/ XỬ LÝ BIẾN VỚI CONTROLLER VÀ TRUYỀN BIẾN CHO VIEW

Mở file mycontroller.php và sữa lại phương thức chaoban như sau:

```
public function chaoban()
{
    $hoten = "Giang Hào Côn";
    return view("Buoi05.chao", compact("hoten"));
}
```

Bên trên là ta đã tạo biến $hoten và gửi biến này tới trang Views test thông qua compact. Để nhận giá trị biến ta viết lại chao.blade.php như sau:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
        crossorigin="anonymous">
    <title>Chào Bạn</title>
</head>
<body>
    <h1 class="alert alert-primary">LARAVEL chào mừng bạn {{$hoten}} đến với Website này</h1>

<!-- Option 1: Bootstrap Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
        crossorigin="anonymous"></script>
</body>
</html>
```

Ta thấy biến $hoten đã được gửi sang Views thành công.

5.2.5/ XỬ LÝ NHIỀU BIẾN VỚI CONTROLLER VÀ TRUYỀN CHO VIEW

Viết lại nội dung phương thức **chaoban** trong **mycontroller.php** như sau:

```
public function chaoban()
{
    $hoten = "Giang Hào Côn";
    $tuoi  = 50;
    return view("Buoi05.chao", compact("hoten", "tuoi"));
}
```

Viết lại nội dung tập tin **chao.blade.php** như sau:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
        crossorigin="anonymous">
    <title>Chào Bạn</title>
</head>
<body>
    <h1 class="alert alert-primary">Chúc mừng sinh nhật lần thứ {{$tuoi}} của bạn {{$hoten}}</h1>

<!-- Option 1: Bootstrap Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
        crossorigin="anonymous"></script>
</body>
</html>
```

Kết quả hiển thị:

5.3/ FORM REQUEST IN LARAVEL

5.3.1/ Tạo View tại Resources/Views

```
<form method="post" action="pathname">
        {{ csrf_field()}}
        …
        <input type="submit" name="tinh" value=" … ">
</form>
```

5.3.2/ Tạo Action tại Controllers

```
public function Action_Name1() {
      return view('View_Name');
 }
public function Action_Name2(Request $req){
      $Variable =$req->Control_Name;
      if (condition) {
            …
            return view('View_Name', compact('…'));
      }
      else {   return view("View_Name", compact('…'));  }
   }
```

5.3.3/ Tạo Route tại Web.PHP

```
Route::get("pathname", "Controller_Name@Action_Name1");
Route::post("pathname", " Controller_Name@Action_Name2 ");
```

5.4/ Sử dụng Blade Template Engine (BTE) trong Laravel

Tất cả các BTE đều có phần mở rộng là .blade.php và nằm trong thư mục resouces/views. Vậy nên việc tạo mới BTE rất đơn giản. Ví dụ để tạo BTE tên là demo thì ta sẽ tạo một file demo.blade.php ở trong thư mục resources/views và nhập vào nội dung như sau:

```
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <title>Demo BTE</title>
</head>
<body>
    <h2>Sử dụng Blade template engine trong Laravel</h2>
</body>
</html>
```

5.4.1./ Các cú pháp sử dụng trong Blade template engine

a) Blade sử dụng cặp ngoặc {{}} để echo giá trị

Ví dụ: mở view demo.blade.php thêm code sau:

```
Hôm nay là ngày: {{date("d-m-Y")}}
```

b) Sử dụng or để xuất giá trị mặc định.

```
Hello, {{ $name or 'Buzz' }}
```

Nếu biến $name không tồn tại thì sẽ hiện ra Buzz

c) Sử dụng các vòng lặp, câu lệnh điều kiện

Để sử dụng vòng lặp hoặc câu điều kiện if thì ta chỉ việc thêm @ ngay trước câu lệnh (theo cú pháp PHP) và kết thúc bằng @end +tên hàm.

```
@if (count($records) === 1)
    Có 1 sản phẩm
@elseif (count($records) > 1)
    Có nhiều sản phẩm
@else
    Không có sản phẩm
@endif


@for ($i = 0; $i < 10; $i++)
    Giá trị của i:  {{ $i }}<br>
@endfor
```

Bài Tập

Bài 01: Xây dựng View có giao diện sau:

Tính Chu Vi - Diện Tích Hình Chữ Nhật

STT	Hình Chữ Nhật	Thao tác
01	Chiều dài= 3.2cm - Chiều rộng = 13cm	Tính
02	Chiều dài= 13.2cm - Chiều rộng = 6.4cm	Tính

Chu vi = 32.4 Diện tích = 41.6

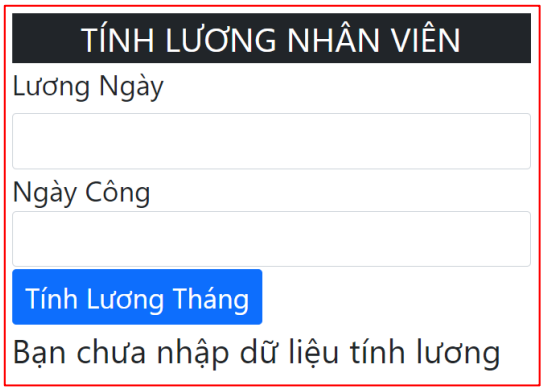
Bài 02: Xây dựng View có giao diện sau:

TÌM NGHIỆM CÁC PHƯƠNG TRÌNH SAU ĐÂY

STT	Phương Trình	Thao Tác
01	3x + 20 = 0	Xem Kết Quả
02	0x + 0 = 0	Xem Kết Quả
03	2x - 9 = 0	Xem Kết Quả
04	0x - 10 = 0	Xem Kết Quả
04	3x - 17 = 0	Xem Kết Quả

Kết Quả Hiện Ở Đây

Bài 03: Xây dựng View có giao diện sau:



Bài 04: Xây dựng View có giao diện sau:

Bài 05: Xây dựng View có giao diện sau:

Bài 06: Xây dựng View có giao diện sau: