

Lecture 6.3

Laravel – Controllers and Views

Course Instructor

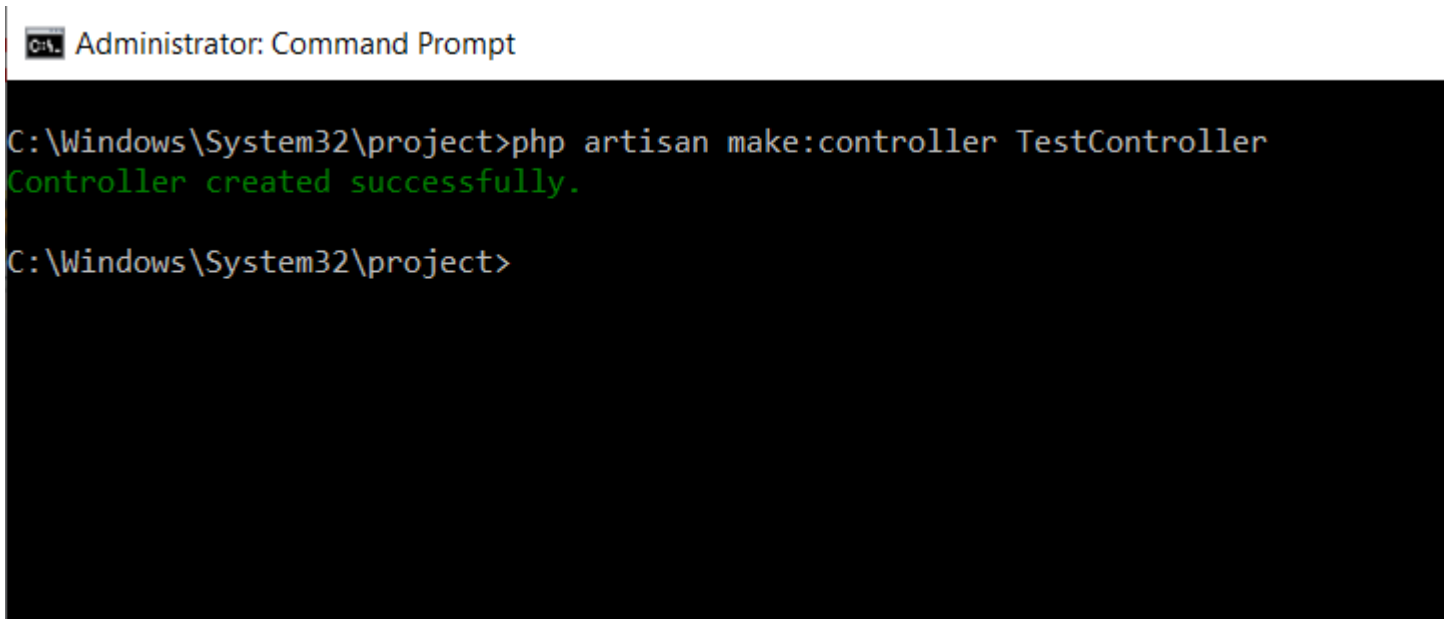
Engr. Madeha Mushtaq

Laravel Controllers

- In Laravel, all Controllers are stored in `app→Http→Controllers` directory.
- Laravel provides a base controller name `Controller.php`
- All laravel controllers should extend the base controller.
- Inside Controller folder, you can create further folders to better organize your controllers.

Laravel Controllers

- Create controller using command:
“php artisan make:controller name”.
- Controller will be saved at \app\Http\Controller.



```
Administrator: Command Prompt
C:\Windows\System32\project>php artisan make:controller TestController
Controller created successfully.
C:\Windows\System32\project>
```

Controller Example

- TestController.php

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class TestController extends Controller
8  {
9      public function Showview()
10     {
11         return view ('hello');
12     }
13 }
14
```

Controller Example

- hello.blade.php

```
<html>
<head>
  <title> Hello View </title>
</head>
<body>
  <h1>hello</h1>
</body>
</html>
```

- Route in web.php

```
Route::get('test', 'TestController@showview');
```

← → ↻ ⓘ 127.0.0.1:8000/test

hello

Output

Passing Data from Controller

- Passing data from controllers to view.
- The task is to get first name and last name, and then passes that full name into a view.
- public function index() {
 \$first = "Alex";
 \$last = "John";
 \$full = \$first . " " . \$last;
 return view('name')→with("fullname", "\$full");
}

Passing Data from Controller

- With() method, it takes two arguments. The first argument is the (key)name, you want to access the variable in the view.
- The second argument is (value)variable that we describe in the function that contain the actual name.

```
with("fullname", "full");
```

- Now open resource → views → name.blade.php and insert this variable {{ \$fullname }} to access the value of the variable.

Passing Data from Controller

- There is another easier way, to pass data to our views from controllers.

return view('name')→withfullname(\$full);

Passing an Array

- Hellocontroller.php

```
1 <?php
2 namespace App\Http\Controllers;
3
4 use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
5 use Illuminate\Foundation\Bus\DispatchesJobs;
6 use Illuminate\Foundation\Validation\ValidatesRequests;
7 use Illuminate\Routing\Controller as BaseController;
8
9 class Hellocontroller extends Controller
10 {
11     public function index()
12     {
13         $subjects = ['Maths', 'English', 'Programming'];
14         return view('hello')->with(['mysub' => $subjects]);
15     }
16 }
17 ?>
```

Passing an Array

- Hello.blade.php

```
1 <html>
2 <head>
3   <title> Hello View </title>
4 </head>
5 <body>
6   <h1>hello</h1>
7   <?php print_r($mysub); ?>
8 </body>
9 </html>
```

- Output:

← → ↻ ⓘ 127.0.0.1:8000/hello

hello

Array ([0] => Maths [1] => English [2] => Programming)

Passing an Array

```
1 <?php
2 namespace App\Http\Controllers;
3
4 use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
5 use Illuminate\Foundation\Bus\DispatchesJobs;
6 use Illuminate\Foundation\Validation\ValidatesRequests;
7 use Illuminate\Routing\Controller as BaseController;
8
9 class Hellocontroller extends Controller
10 {
11     public function index()
12     {
13         $subjects = ['Maths', 'English', 'Programming'];
14         $marks = [50,40,45];
15         return view ('hello')-> with (['mysub' => $subjects, 'marks' => $marks]);
16     }
17 }
18 ?>
```

Alternatively, we can also use this:

```
$subjects = ['Maths', 'English', 'Programming'];
$marks = [50,40,45];
return view ('hello')-> withmysub($subjects)-> withmarks($marks);
```

Passing an Array

```
1 <html>
2 <head>
3   <title> Hello View </title>
4 </head>
5 <body>
6   <h1>hello</h1>
7   <?php print_r($mysub); ?> <br>
8   <?php print_r($marks); ?>
9 </body>
10</html>
```

← → ↻ ⓘ 127.0.0.1:8000/hello

hello

Array ([0] => Maths [1] => English [2] => Programming)

Array ([0] => 50 [1] => 40 [2] => 45)

Views

- In Laravel, all views are stored in views folder inside resource folder.
- We can create our own folders inside view folder to better organize views.
- By default, Laravel provide us welcome.blade.php
- View contains Php, plain HTML(CSS+JavaScript+Bootstrap) + Blade.

Blade Templating Engine

- Blade is a templating engine inside laravel.
- Views are always stored with extension `.blade.php`
- The `.blade.php` extension instructs the framework to use the blade templating engine to render the view.
- You may use plain php templates with Laravel.
However, blade provides convenient short-cuts for writing clean, terse templates.

Blade Templating Engine

- PHP Syntax:

```
<?php foreach($customers as $customer) ?>
```

- Blade syntax:

```
@foreach($customers as $customer)
```

```
@endforeach
```

Blade Templating Engine

- PHP Syntax:

<p><?php echo \$customer→cname; ?></p>

- Blade Syntax:

<p>{{ \$customer→cname }}</p>

Blade Templating Engine

- PHP Syntax
 - `<?php if(true):`
 `echo 'hello';`
 `endif; ?>`
- Blade Syntax
 - `@if(true)`
 `{{ 'hello' }}`
 `@endif`

Comments in Blade Templates

- Comments in blade begin with `{{--` and end with `--}}`.
- They can span multiple lines.

Layouts

- Layouts are often the starting point of many web development projects.
- Layouts are designs through which we can separate repetitive portion of a website.
- The layout thus designed can be used by other views, and includes a consistent design and structure.

- about.blade.php

```
@extends('main')
@section('stylesheets')
<link rel="stylesheet"
href="main.css" type="text/css"
/>
@endsection

@section('content')
<h1> About Us Page</h1>
<p> This is content for About Us
page</p>
@endsection
```

- services.blade.php

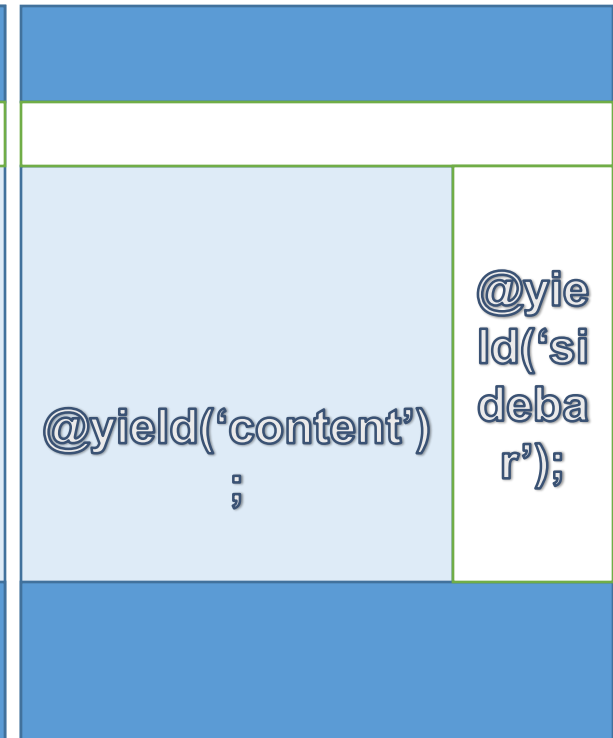
```
@extends('main1')
@section('stylesheets')
<link rel="stylesheet" href="main.css"
type="text/css" />
@endsection
@section('content')
<h1> About Us Page</h1>
<p> This is content for About Us
page</p>
@endsection
@section('sidebar')
<ul><li>list of services</li></ul>
@endsection
```

Layouts

main.php



main1.blade.php



Layouts

- `@yield('content')` this is basically we call a layout section, `content` is the title given to it, so this is container we leave that blank, its contents are dynamically filled in.

Example

- Create a folder Layouts, create Master.blade.php in it.

```
1 <html>
2 <head>
3   <title> Master Layout </title>
4 </head>
5 <body>
6   <h3>Hello, in Master</h3>
7   @yield('body')
8 </body>
9 </html>
```

Example

- Marks_sub.blade.php

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

@section('body')
<h1>hello</h1>
<?php print_r($mysub); ?> <br>
<?php print_r($marks); ?>
@endsection
```

Different ways to print data using Blade

- `{{ $data }}` is equivalent to this in Php:
 - `<?php echo $data; ?>`
- `@{{ $data }}` (It will display `{{ $data }}` as it is).
- `{!! $data !!}` (It will run the script if any passed from the controller).

Conditional Coding Using Blade

- condition.blade.php

```
1 @extends ('Layouts/master')
2
3 @section('body')
4 {{isset($data)? $data: 'Data not Found'}}
5 @endsection
6
```

Alternatively, it will work the same way.

```
{{ $data or 'Data not Found' }}
```

Example

- conditionalcontroller.php

```
1  <?php
2  namespace App\Http\Controllers;
3
4  use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
5  use Illuminate\Foundation\Bus\DispatchesJobs;
6  use Illuminate\Foundation\Validation\ValidatesRequests;
7  use Illuminate\Routing\Controller as BaseController;
8  use App\student;
9  use App\Teacher;
10 use Illuminate\Http\Request;
11
12 class conditionalcontroller extends Controller
13 {
14     public function index()
15     {
16         $Text = "Hello";
17         return view('Condition') ->with ('data',$Text);
18     }
19 }
20 ?>
21
```

Example

- condntion.blade.php

```
1  @extends ('Layouts/master')
2
3  @section('body')
4  <? php $u = "Data not found"; ?>
5  @if($data == "Hello")
6      {{$data}}
7  @elseif($data == "Hi")
8      {{$data}}
9  @else
10     {{$u}}
11 @endif
12
13 @endsection
```