Web Technologies
Lecture#26

Lecture Outline

1. Creating Controllers
2. Loading view from controller
3. Data Passing: Controller to View
4. Blade syntaxes in views
5. Data Passing: Views to Controller
6. Advanced Routing

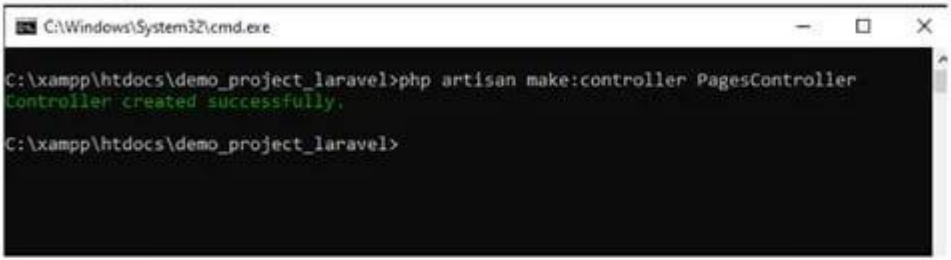
Creating Controllers

Topic sub heading..

- Instead of defining all the requests in route closure its better to group them in another place. Its more organized.
- Controllers can group related request handling logic into a single class.
- Controllers are stored in the `app/Http/Controllers` directory.
- Controller extends the base controller class included with Laravel.
- The base class provides a few convenience methods such as the middleware method, which may be used to attach middleware to controller actions.
- To create controller you can go to `app/Http/Controllers` and create a controller file with `.php` extension.
- You need to write a `class` of your own.
- Your `controller class` should extend the default `controller` class.
- Laravel also provides an artisan command that automatically creates a controller for you. (Open cmd in project and run the command)

```
php artisan make:controller your_controller_name
php artisan make:controller PagesController
```

Creating Controllers



PagesController.php

app > Http > Controllers > PagesController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    //
}

```

Loading View from Controller

- Previously we were loading views from the route directly with `Closure`. Now we will bind controllers method with route Closure and load the view.
- Write a function in Controller.

```
function Hello(){
        return view("hello");
}
```

- This function will return the view residing in `resources/views/hello.blade.php`
- Now to bind the function with route(go to `web.php`)

```
Route::method('route','ControllerName@ControllerFunction');[General Syntax]
Route::get('/hello','PagesController@hello');
```

- So the steps are
  - Create a controller in `app/Http/Controllers/`
  - Create a view in `resources/views/ with .blade.php` extension
  - Write a function in controller to load the view
  - Create a route in `routes/web.php` and bind the controller function

Loading Views from Controller

PagesController.php

app > Http > Controllers > PagesController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    function Hello(){
        return view("hello");
    }
}
```

Controller

hello.blade.php

resources > views > hello.blade.php > html

```html
<html>
    <head></head>
    <body>
        <h1>Hello Advanced Web</h1>
    </body>
</html>
```

Views

web.php

routes > web.php

```php
<?php

Route::get('/', function () {
    return view('welcome');
});

Route::get('/hello','PagesController@hello')->name('hello');
```

Route

Output

Data Passing: Controller to View

- As most of the cases controller functions will be responsible for database operations and calculations, we will be needing to pass data from controller to views to build dynamic web applications.
- Values can be passed with an associative array and the passed values can be retrieved with the key in blade files.
  - `return view('filename', ['key' => 'Value' , 'key' => 'Value']); //Passing value`
  - `<?php echo $key; ?> //retrieving in blade`
- As an alternative to passing a complete array of data to the `view` helper function, you may use the `with` method to add individual pieces of data to the view.
  - `return view('filename')->with('blade_key', 'value'); //Passing value`
- You may display the contents of the variable like so

```
Hello, {{ $name }}.
      Instead of using
Hello, <?php echo $name; ?>
```

By default, Blade `{{ }}` statements are automatically sent through PHP's `htmlspecialchars` function

Passing Data to View [Associative Array]

PagesController.php

app > Http > Controllers > PagesController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    function Hello(){
        $name="Jhon";
        $profession="Teacher";
        $address="Bangladesh";
        return view("hello",
        [
            "name"=>$name,
            "prof"=>$profession,
            "address"=>$address
        ]);
    }
}
```

Passing the value

hello.blade.php

resources > views > hello.blade.php > html

```html
<html>
    <head></head>
    <body>
        <h1>Hello Advanced Web </h1>
        <div>
            Name: <?php echo $name;?> <br>
            Profession: <?php echo $prof;?> <br>
            Address: <?php echo $address;?> <br>
        </div>
    </body>
</html>
```

Using them in views

localhost:8000/hello

Hello Advanced Web

Name: Jhon
Profession: Teacher
Address: Bangladesh

Output

Passing Data to View [with function]

PagesController.php

app > Http > Controllers > PagesController.php

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class PagesController extends Controller
{
    function Hello(){
        $name="Jhon";
        $profession="Teacher";
        $address="Bangladesh";
        return view("hello")
        ->with("name",$name)
        ->with("prof",$profession)
        ->with("address",$address);
    }
}
```

Passing the value

hello.blade.php

resources > views > hello.blade.php > html

```blade
<html>
    <head></head>
    <body>
        <h1>Hello Advanced Web </h1>
        <div>
            Name: {{$name}} <br>
            Profession: {{$prof}} <br>
            Address: {{$address}} <br>
        </div>
    </body>
</html>
```

Shortcut to echo variable

localhost:8000/hello

Hello Advanced Web

Name: Jhon
Profession: Teacher
Address: Bangladesh

Output

Blade Syntaxes in Views

Topic sub heading..

- Echo data

```
{{ $variable }}
```

- If else

```
@if (condition)
    Statement
@elseif (condition)
    Statement
@else
    Statement
@endif
```

- For loop

```
@for($i=0; $i<$length; $i++)
    Staement
@endfor
```

- Foreach loop

```
@foreach ($array as $element)
    {{$element->property}}
@endforeach
```

- While loop

```
@while (condition)
    Statement
@endwhile
```

- Including other views

```
@include('view')
```

- Defining master layout and extending those will be discussed later.

Passing array to blade Syntax in blade

PagesController.php

app > Http > Controllers > PagesController.php

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class PagesController extends Controller
{
    function Hello(){
        $ages= ["Rahim"=>"34","karim"=>"36"];
        return view("hello")
        ->with("ages",$ages);
    }
}
```

Passing array

hello.blade.php

resources > views > hello.blade.php > html > body > div

```blade
<html>
    <head></head>
    <body>
        <h1>Hello Advanced Web </h1>
        <div>
            @foreach($ages as $person=>$age)
              {{$person}} {{$age}} <br>
            @endforeach
        </div>
    </body>
</html>
```

Usage of foreach loop

Output

If else syntax in blade

PagesController.php

app > Http > Controllers > PagesController.php

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class PagesController extends Controller
{
    function Hello(){
        $ages= ["Rahim"=>"34","karim"=>"36","Salam"=>"17"];
        return view("hello")
        ->with("ages",$ages);
    }
}
```

Passing array

hello.blade.php

resources > views > hello.blade.php > html

```blade
<html>
    <head></head>
    <body>
        <h1>Hello Advanced Web </h1>
        <div>
            @foreach($ages as $name=>$age)
            @if($age > 18)
              {{$name}} is a voter <br>
            @else
              {{$name}} is under aged <br>
            @endif
            @endforeach
        </div>
    </body>
</html>
```

If else

Output

Passing object in blade and Usage

Person class

Person objects

Person objects

PagesController.php

app > Http > Controllers > PagesController.php

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Person{
    var $age;
    var $name;
    function __construct($name,$age){
        $this->age = $age;
        $this->name = $name;
    }
}
class PagesController extends Controller
{
    function Hello(){
        $p1 = new Person("Karim","34");
        $p2 = new Person("Rahim","36");
        $p3 = new Person("Salam","17");
        $persons=array($p1,$p2,$p3);
        return view("hello")
        ->with("persons",$persons);
    }
}
```

hello.blade.php

resources > views > hello.blade.php > html

```
<html>
    <head></head>
    <body>
        <h1>Hello Advanced Web </h1>
        <div>
            @foreach($persons as $person)
              @if($person->age > 18)
                {{$person->name}} is a voter <br>
              @else
                {{$person->name}} is under aged <br>
              @endif
            @endforeach
        </div>
    </body>
</html>
```

localhost:8000/hello

Hello Advanced Web

Rahim is a voter
karim is a voter
Salam is under aged

Output

Data Passing: View to Controller

- In web user input data can be passed to server in 2 ways.
  - Form data [Get or Post]
  - URL parameter [Get Request]
- Create a form in `resources/views/registration.blade.php`
- Create route in `routes/web.php` and function in `PagesController.php`
- Don't forget put `{{csrf_field()}}` inside form
- Create a post route for Submitting the form and function

```
function register(Request $request){
    return "Submitted";
}
```

- The function should receive a `Request` object which holds all the submitted value.
  - The submitted values are the properties of `$request` object.

Passing value from view to Controller

registration.blade.php

```
<html>
    <head></head>
    <body>
        <form action="/register" method="post">
            {{csrf_field()}}
            Name: <input type="text" name="name"><br>
            Profession:<input type="text" name="profession"><br>
            Address: <input type="text" name="address"><br>
            <input type="submit" name="submit">
        </form>
    </body>
</html>
```

PagesController.php

```
function registration(){
    return view("registration");
}
function register(Request $request){
    $output = "<h1>Submitted</h1>";
    $output.="Name: ".$request->name;
    $output.="<br>Profession: ".$request->profession;
    $output.="<br>Address: ".$request->address;
    return $output;
}
```

```
Route::get('/registration','PagesController@registration');
Route::post('/register','PagesController@register');
```

localhost:8000/registration

Name: Jhon
Profession: Teacher
Address: Bangladesh
Submit

Output

localhost:8000/register

Submitted

Name: Jhon
Profession: Teacher
Address: Bangladesh

Output

Advanced Routing

- Through routing you can also pass dynamic values to server.

```
Route::get('/user/{id}', function($id)
{
    return "<b>The passed id is ".$id."</b>";
});
```

Parameter

Argument

- The value inside {} is dynamic and should be caught in function parameter to use.

110 is passed

The passed id is 110

- You can also create like this

```
Route::get('/user/{id}/files/{file_id}',function($id,$fileid){
    return "<b>The passed id is $id and file id
});
```

The passed id is 110 and file id is 12

Routing parameters

- should consist of alphabetic characters, and may not contain a - character. Instead of using the - character, use an underscore (_).
- Previously shown parameters are mandatory parameters you must pass those parameters otherwise it will get an error.
- Optional Route parameters

```
Route::get('/user/{id?}', function($id = null)
{
    return "<b>The passed id is ".$id."</b>";
});
```

- Optional Route parameters with default value

```
Route::get('/user/{id?}', function($id = 'Default')
{
    return "<b>The passed id is ".$id."</b>";
});
```

- Routes can have names to access them from the codes.

```
Route::get('/hello',function(){
    return "Hello World";
})->name('hello');
```

```
Route::get('/user/{id?}',function($id = 'Default'){
    return redirect()->route('hello');
    return "<b>The passed id is ".$id."</b>";
});
```

Access with name

MUST READ: https://laravel.com/docs/4.2/routing

Books

- PHP Advanced and Object-Oriented Programming, 3rd Edition; Larry Ullman; Peachpit, Press, 2013
- PHP Objects, Patterns and Practice, 5th Edition; Matt Zandstra; Apress, 2016
- Learning PHP, MySQL, JavaScript and CSS, 2nd Edition; Robin Nixon; O'Reilly, 2009
- Eloquent JavaScript: A Modern Introduction to Programming; Marijn Haverbeke; 2011
- Learning Node.js: A Hands On Guide to Building Web Applications in JavaScript; Marc Wandschneider; Addison-Wesley, 2013
- Beginning Node.js; Basarat Ali Syed; Apress, 2014

References

1. https://www.w3adda.com/laravel-tutorial/laravel-blade-template
2. https://laravel.com/docs/7.x/session
3. https://laravel.com/docs/7.x/controllers
4. https://laravel.com/docs/7.x/views
5. https://laravel.com/docs/7.x/blade


AMERICAN INTERNATIONAL UNIVERSITY
PRAESIDIUM
1994
BANGLADESH

Thank You!