Routing & Controller

Introduction to

In this practicum, you will learn about routing and controllers in a web framework.

Learning Objectives

1. Students understand the concept of Web Framework routing
2. Students implement routing on the Web Framework
3. Students understand the concept of a controller
4. Students implement a controller on the Web Framework

Tools and materials

1. PC or Laptop
2. Text Editor / IDE (PHPStorm or VSCode recommendation)
3. Web Browser
4. PHP

Introduction to Routing

If we freely translated, route means path. And we can imagine that as a routing concept in Laravel, namely the URL paths that can be accessed by application users and where it is processed. For example, there is a simple / hello route, this route can be accessed at http://localhost:8000/ hello and will display the string 'Hello World'. The basic syntax of routing is as follows:

```
Route::verb("/path", callback);
```

To create a route, a Facade Route is used followed by a verb which is an HTTP verb, generally consisting of get, post, put, delete, options, patch. In addition, it requires a path in the form of a URL after the application domain name accessed by the user. And at the end there is a callback which can be a callback function or a controller action that executes logic when the path is accessed by the user.

Here is an example of routing using a callback function which will display the message 'Hello World'.

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

The route above can be translated when the user accesses the URL at http: // localhost: 8000 / hello will execute a callback function which displays the message 'Hello World'. Users will get the message which will be displayed on their browser. However, the use of callback functions is rarely used in real application development, because the complex logic makes code difficult to maintain. As a solution the controller concept was introduced. If the route d above is converted to a controller it will be as follows:

```
Route::get('/hello', [WelcomeController::class,'hello']);
```

Using a controller simplifies the definition of a route, as it is sufficient to write the controller name followed by the action on the callback. The technical implementation is in the WelcomeController class.

Route definitions should be written according to their purpose. In general, laravel divides into four places, namely:

1. `routes / web.php` is used for web standards
2. `routes / api.php` is used for web service / API
3. `routes / console.php` is used for the command line
4. `routes / channel.php` is used to broadcast channels via websocket

In general, the application that is created is simply routes / web.php and routes / api.php. Even if the app doesn't need to provide an API, just use routes / web.php. For more documentation you can read https://laravel.com/docs/8.x/routing the official Laravel documentation.

Basic Routing

Basically Routing in Laravel requires information about the http verb, then input the url and what to do when receiving the url. To create a route you can use a callback function or use a controller. Here is an example of a route creation plan and its implementation:

No	Http Verb	Url	Fungsi
1	get	/hello	Tampilkan String Hello ke browser.
2	get	/world	Tampilkan String

			World ke browser

The following is an example of routing for case number 1 using a callback function

```
use Illuminate\Support\Facades\Route;

Route::get('/hello', function () {
    return 'Hello';
});
```

Pay close attention to what type of route to use then input the url to enter and how to add a callback function. As for the second case, if you want to solve it using a controller function, you have to create a controller that contains the methods you mention on the router. The following is the program code for creating a route using the controller

```
use Illuminate\Support\Facades\Route;

Route::get('/world', [WelcomeController::class,'world']);
```

Notice that on routes with controllers, the only difference is in replacing function callbacks with controller parameters and naming the functions. Understand that in Laravel you have to list all accessible urls in this routes file.

Router Method

In Laravel you can use any http verb to be installed as the router method you want to use, it has been explained earlier that all http verbs can be served by a router on laravel. The endpoint / url of the router should follow the following best practice where a resource can be served with a different function for each http verb.

Resource	POST	GET	PUT	DELETE
/mahasiswa	Membuat record mahasiswa baru	Mengambil Daftar Mahasiswa	Update banyak data mahasiswa	Delete banyak data mahasiswa
/mahasiswa/{id}	Error	Tampilkan Data Satu Mahasiswa	Update data mahasiswa jika ada data dengan id yang dikirim	Delete satu data mahasiswa

Look at the table above, creating a route table like this is better than creating the existing routing table. In this way, it is clearer that a resource / url (/ student) has any http verb and what is done with each http verb. Note that laravel can support a route that has more than one http verb or has all http verbs. The following is the routing program code for the table above

```
Route::get('mahasiswa', function ($id) {
});

Route::post('mahasiswa', function ($id) {
});

Route::put('mahasiswa', function ($id) {
});

Route::delete('mahasiswa', function ($id) {
});

Route::get('mahasiswa/{id}', function ($id) {
});

Route::put('mahasiswa/{id}', function ($id) {
});

Route::delete('mahasiswa/{id}', function ($id) {
});
```

You can check and validate whether the route created is correct by using the following commands

```
php artisan route:list
```

The output of the command if the routing you made is correct will be like this

Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	/		Closure	web
	GET|HEAD	api/user		Closure	api
					auth:api
	GET|HEAD	mahasiswa		Closure	web
	POST	mahasiswa		Closure	web
	PUT	mahasiswa		Closure	web
	DELETE	mahasiswa		Closure	web
	GET|HEAD	mahasiswa/{id}		Closure	web
	PUT	mahasiswa/{id}		Closure	web
	DELETE	mahasiswa/{id}		Closure	web

If you need a route that can have more than one http the routing method can be created this way.

```
Route::match(['get', 'post'], '/specialUrl', function () {
});

Route::any('/specialMahasiswa', function ($id) {
});
```

```
+--------+--------------------------------------+-----------------+------+---------+------------+
| Domain | Method                               | URI             | Name | Action  | Middleware |
+--------+--------------------------------------+-----------------+------+---------+------------+
|        | GET|HEAD                             | /               |      | Closure | web        |
|        | GET|HEAD                             | api/user        |      | Closure | api        |
|        |                                      |                 |      |         | auth:api   |
|        | GET|HEAD|POST|PUT|PATCH|DELETE|OPTIONS | specialMahasiswa |      | Closure | web        |
|        | GET|POST|HEAD                         | specialUrl      |      | Closure | web        |
+--------+--------------------------------------+-----------------+------+---------+------------+
```

Dependency Injection

Routes in laravel can be done with dependency injection where a route can be assigned the required dependencies, Laravel will resolve this dependency automatically and send it in the callback function of the route. For example, on the route below, we leave a dependency Request on the / users route

```
use Illuminate\Http\Request;

Route::get('/users', function (Request $request) {
    // ...
});
```

CSRF Protection

All routes defined in web.php must include a CSRF Token for processing the http verb POST, PUT, PATCH or DELETE if it does not have a CSRF token, the request will be rejected. Here's how to add CSRF to an existing form in the html template

```
<form method="POST" action="/profile">
    @csrf
    ...
</form>
```

Redirect Routes

To redirect to laravel, you can use Route :: redirect, how to use it, can be seen in the program code below.

```
Route::redirect('/here', '/there');
```

You will often use this redirect in CRUD cases or other cases that require a redirect

View Routes

Laravel also provides a special route that makes it easy for you to create a route without using a controller or callback function. These routes take the input in the form of a url and return a view. Here's how to create view routes.

```
Route::view('/welcome', 'welcome');

Route::view('/welcome', 'welcome', ['name' => 'Taylor']);
```

Notice that the routes above / welcome will display the welcome view and the second route / welcome will display the welcome view with additional data in the form of a name variable.

Route Parameters

In some cases you will need one of the segments of the url as a mandatory parameter for the controller or routing callback function. For example you need a user id sent via a URL, to make the routing can be done in the following way

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

Route::get('/posts/{post}/comments/{comment}', function ($postId,
$commentId) {
    //
});
```

Pay attention to how to write parameters and how to use parameters in function / controller callbacks. Parameters are written using curly braces {} and the corresponding variables are arranged as callback param in the callback function. Note the order, this order also applies to routes that use the controller function.

If a route requires mandatory parameters but also requires dependency from dependency injection, all parameters are written after the dependency is written, here is an example of a route.

```
Route::get('/user/{id}', function (Request $request, $id) {
    return 'User '.$id;
});
```

As for parameters that are optional or parameters that are not always on the route, you can use optional parameters in the route you created. Here's how to create optional param in routing.

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

Route::get('/user/{name?}', function ($name = 'John') {
    return $name;
});
```

Notice in the function above you can see that an optional parameter is marked ? and the initial value can be made in the callback function or the controller function.

Route Name

A route can be given a custom name to make it easier to generate urls when coding. An example of naming routing can be seen in the following program code.

```
Route::get('/user/profile', function () {
    //
})->name('profile');

Route::get(
    '/user/profile',
    [UserProfileController::class, 'show']
)->name('profile');

// Generating URLs...
$url = route('profile');

// Generating Redirects...
return redirect()->route('profile');
```

Route Group

Several routes that have the same attributes as the same middleware can be grouped into one group to make it easier to write routes. Besides being used for middleware, there is still a route group for routes that are under one subdomain. An example of using the route group is as follows:

```
Route::middleware(['first', 'second'])->group(function () {
    Route::get('/', function () {
        // Uses first & second middleware...
    });

    Route::get('/user/profile', function () {
        // Uses first & second middleware...
    });
});

Route::domain('{account}.example.com')->group(function () {
    Route::get('user/{id}', function ($account, $id) {
        //
    });
});
```

Route prefix

Route grouping can also be done for routes that have the same prefix, an example of creating a route with the prefix can be seen in the program code below

```
Route::prefix('admin')->group(function () {
    Route::get('/users', function () {
        // Matches The "/admin/users" URL
    });
});
```

Controller

Controllers are used to organize the application logic into a more structured manner. Any related application action logic can be collected in a single Controller class. Or a Controller can contain only one action. Controllers in Laravel are stored in the app / Http / Controllers folder.

To create a controller in Laravel, a command is provided to generate the basic structure. You can use the artisan command followed by the definition of the controller name to be created. Please pay attention to the following command examples:

```
php artisan make:controller WelcomeController
```

Please open the file at app / Http / Controllers / WelcomeController.php. The structure of the controller can be described as follows:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WelcomeController extends Controller
{
      //
}
```

To define an action, please add a function with public access. So that the controller above becomes as follows:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WelcomeController extends Controller
{
      public function hello() {
          return 'Hello World';
      }
}
```

After a controller action has been defined, you can then add the controller to the route. So the route is as follows:

```
Route::get('/hello', 'WelcomeController@hello');
```

For more complete documentation regarding controllers, you can read the official documentation from Laravel at the https://laravel.com/docs/8.x/controllers link.

Resource Controller

Especially for controllers that are connected to the Eloquent model and CRUD operations can be performed on the Eloquent model, a controller of type Resource Controller can be created where by creating a controller resource you will create a complete controller with method methods that support CRUD processes and a resource route. that holds the route for that controller. To make it done by running the following command in the terminal.

```
php artisan make:controller PhotoController --resource
```

This command will generate a controller named PhotoController which contains standard methods for CRUD processing.

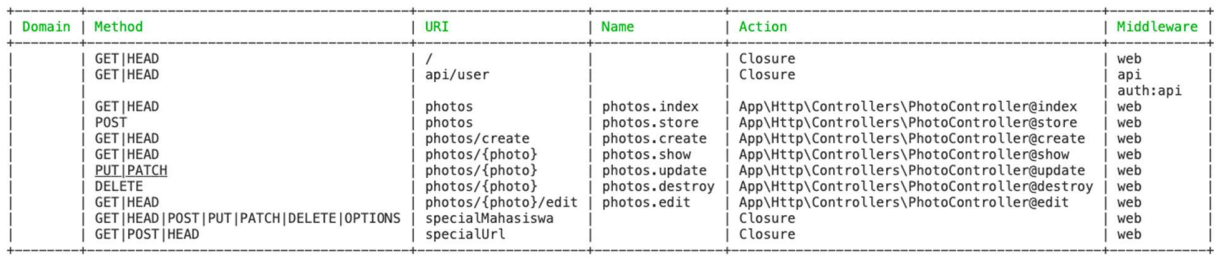
```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PhotoController extends Controller
{
      /**
      * Display a listing of the resource.
      *
      * @return \Illuminate\Http\Response
      */
      public function index()
      {
      //
      }

      /**
      * Show the form for creating a new resource.
      *
      * @return \Illuminate\Http\Response
      */
      public function create()
      {
      //
      }

      /**
      * Store a newly created resource in storage.
      *
      * @param  \Illuminate\Http\Request  $request
      * @return \Illuminate\Http\Response
      */
```

```
    public function store(Request $request)
    {
    //
    }

    /**
    * Display the specified resource.
    *
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
    public function show($id)
    {
    //
    }

    /**
    * Show the form for editing the specified resource.
    *
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
    public function edit($id)
    {
    //
    }

    /**
    * Update the specified resource in storage.
    *
    * @param  \Illuminate\Http\Request  $request
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
    public function update(Request $request, $id)
    {
    //
    }

    /**
    * Remove the specified resource from storage.
    *
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
    public function destroy($id)
    {
    //
    }
}
```

After the controller is successfully generated, a route must be created so that it can be connected to the frontend, add the following program code to the web.php file.

```
Route::resource('photos', PhotoController::class);
```

If now run check list route, the following route will be generated



Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	/		Closure	web
	GET|HEAD	api/user		Closure	api auth:api
	GET|HEAD	photos	photos.index	App\Http\Controllers\PhotoController@index	web
	POST	photos	photos.store	App\Http\Controllers\PhotoController@store	web
	GET|HEAD	photos/create	photos.create	App\Http\Controllers\PhotoController@create	web
	GET|HEAD	photos/{photo}	photos.show	App\Http\Controllers\PhotoController@show	web
	PUT|PATCH	photos/{photo}	photos.update	App\Http\Controllers\PhotoController@update	web
	DELETE	photos/{photo}	photos.destroy	App\Http\Controllers\PhotoController@destroy	web
	GET|HEAD	photos/{photo}/edit	photos.edit	App\Http\Controllers\PhotoController@edit	web
	GET|HEAD|POST|PUT|PATCH|DELETE|OPTIONS	specialMahasiswa		Closure	web
	GET|POST|HEAD	specialUrl		Closure	web

Notice in the route list all the related routes for crud photo have been generated by laravel. If not all of the routes in the controller resource are needed, you can reduce them by updating the routes in web.php to be like this

```
Route::resource('photos', PhotoController::class)->only([
    'index', 'show'
]);

Route::resource('photos', PhotoController::class)->except([
    'create', 'store', 'update', 'destroy'
]);
```

Note that we can use only or except keywords according to the needs of the controller that is created.

Praktikum

Praktikum 1 - Routing Web Framework Laravel

1. Create a new laravel project named 02_praktikum_web_lanjut_satu
2. Create a routing for urls with the following requirements, use a callback function in the form of an anonymous function to output the output.

```
Route::get('/', function ($id) {
      echo "Selamat Datang";
});
```

URL	**Output**
/	Display the message 'Welcome'
/about	Displays your NIM and name
/articles/{id}	Showing Output "Article Pages with ID {id}" replace the id according to the input from the url

3. Use the parameter route concept so that it can handle requests to URLs with the /articles/1, /articles/2 pattern. Show the parameter id so that when the URL is accessed it displays the message 'Article page with id 1' if entered is 1.
4. Save this practicum in your project using Git publish to github with the name 02_praktikum_web_lanjut_satu to your github repository.

Praktikum 2 - Controller Web Framework Laravel

1. Create a new project named 02_praktikum_web_lanjut_dua
2. Modification of Practicum 1 with the concept of a controller. Move the execution logic into a controller named PageController. Save these changes using Git.

Resource	POST	GET	PUT	DELETE
/		Show a 'Welcome' Message PageController: index		
/about		Show Name and NIM PageController: about		
/articles/{id}		Show dynamic		

		page ‘Article Pages with Id {id}’ id replaced according to input from url PageController: articles		

```
Route::get('/', [PageController::class, 'index']);
```

3. Modify the previous implementation with the Single Action Controller concept. So that for the final results you get, you have a HomeController, AboutController and ArticleController. Delete the route to PageController in your project. Save these changes using Git.

```
Route::get('/', [HomeController::class, 'index']);
```

4. Save this practicum in your project using Git publish to github with the name 02_praktikum_web_luntung_dua to your github repository.

Praktikum 3 - Desain Routing Web Company Profile

A company asks you to create a company profile using Laravel, this company requires the following features:

1	Home page Displays the start page of the website
2	Products page Displays a list of products /category/marbel-edu-games /category/marbel-and-friends-kids-games /category/riri-story-books /category/kolak-kids-songs
3	News page Displays List of news /news /news/educa-studio-berbagi-untuk-warga-sekitar-terdampak-covid-19

4	Program page Displays the list of program /program/karir /program/magang /program/kunjungan-industri
5	About Us page Displays About Us /about-us
6	Contact Us page Displays Contact Us /contact-us (Can Post contacts to backend)

Website Reference: https://www.educastudio.com/

Create a route for this website using the concept of routing on laravel using route group, route with parameters, route with controller and others.