**ADVANCED WEB PROGRAMMING**

**ROUTING, CONTROLLER, AND VIEW**



**KHEN MUHAMMAD CAHYO**

[**244107023002**](mailto:%32%34%34%310%37%302%33%3002@%70%6f%6cine%6d%61%2e%61%63%2e%69%64)

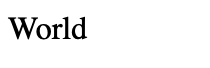
**TI-2I**

**STATE POLYTECHNIC OF MALANG**

**INFORMATION TECHNOLOGY DEPARTMENT**

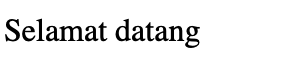
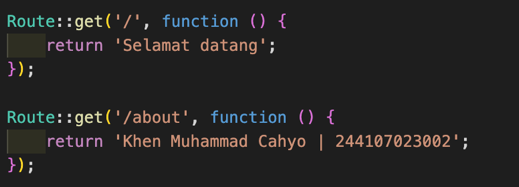
**PRACTICUM ROUTING**

* **Basic Routing (b-e)**



The code defines two routes in a Laravel application using the Route::get method, which handles HTTP GET requests. The first route listens for requests at the /hello URL and returns the string "Hello World", while the second route listens at /world and returns the string "World". These routes demonstrate how Laravel can handle multiple endpoints by mapping URLs to closures that return simple responses without requiring a controller.

* **Basic Routing (f-g)**



* **Route Parameters (a-b)**



The code defines a dynamic route in Laravel using Route::get, which listens for HTTP GET requests at the /user/{name} URL. The {name} part is a route parameter, meaning any value passed in that position will be captured and assigned to the $name variable. When the route is accessed (e.g., /user/KhenCahyo), Laravel will execute the provided closure and return the string "Nama saya KhenCahyo". This demonstrates how Laravel handles route parameters to create dynamic and customizable URLs.

* **Route Parameters (c)**



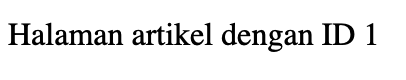
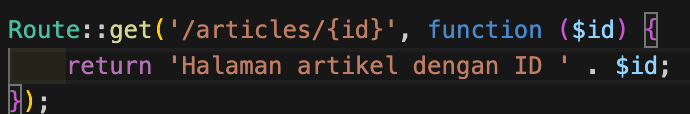
When we try to access /user without providing a {name} parameter, Laravel will return a 404 Not Found error page. This happens because the route is defined to require a {name} parameter, and without it, Laravel does not match the request to any defined route.

* **Route Parameters (d-e)**

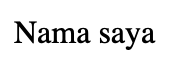


The code defines a Laravel route that handles HTTP GET requests for URLs following the pattern /posts/{post}/comments/{comment}. It includes two route parameters: {post} and {comment}, which capture dynamic values from the URL. These values are then passed as arguments $postId and $commentId to the closure function. When accessed with a URL like /posts/3/comments/7, Laravel assigns 3 to $postId and 7 to $commentId, returning the response "Pos ke-3 Komentar ke-: 7". This demonstrates how Laravel supports multiple route parameters to handle nested resources dynamically. However, to ensure the function works correctly, the parameters inside the closure should match the route placeholders, meaning $post and $comment should be used instead of $postId and $commentId.

* **Route Parameters (f)**

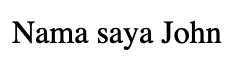
****

* **Optional Parameters (a-c)**

****

When visiting /user with no name provided, the {name?} parameter is optional, so the $name variable defaults to null, and the response will be "Nama saya " without any name. However, when visiting /user/Khen Cahyo, the {name} parameter is supplied with the value "Khen Cahyo", and the closure receives that value in the $name variable, resulting in the response "Nama saya Khen Cahyo". If you want to handle the case where no name is provided more gracefully, you can set a default value for $name, such as "Guest", so the response would be "Nama saya Guest" when accessing /user without a name.

* **Optional Parameters (d-e)**

****

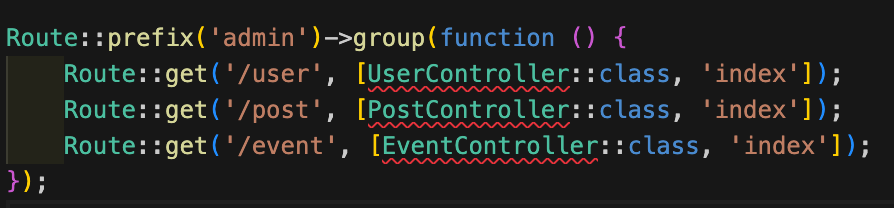
The code defines a Laravel route that listens for HTTP GET requests at /user/{name?}, where {name?} is an optional route parameter. The $name variable in the closure has a default value of 'John', so if no name is provided in the URL (e.g., /user), the response will be "Nama saya John". If a name is provided (e.g., /user/Khen Cahyo), the route will use the provided name, and the response will be "Nama saya Khen Cahyo". This setup ensures that a default name is used when the parameter is not given, providing a more user-friendly fallback.

* **Route Name (a-b)**

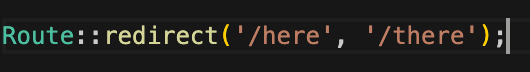
****

The code defines a Laravel route that listens for HTTP GET requests at the /user/profile URL and associates it with a named route profile using the name('profile') method. The route currently does not return any response because the closure is empty. When trying to visit the URL /profile, it will not match this route, as it is specifically defined for /user/profile. If you want to access the route with /profile, you would need to change the URL pattern in the route definition to /profile or adjust your link to point to /user/profile.

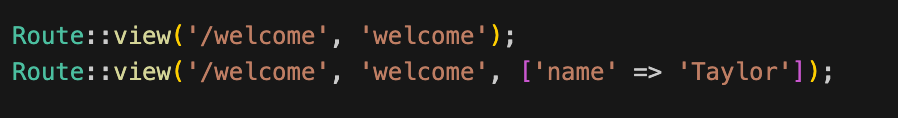
* **Route Group and Route Prefixes**

****

* **Route Redirect**

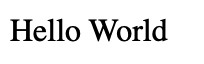
****

* **View Routes**

****

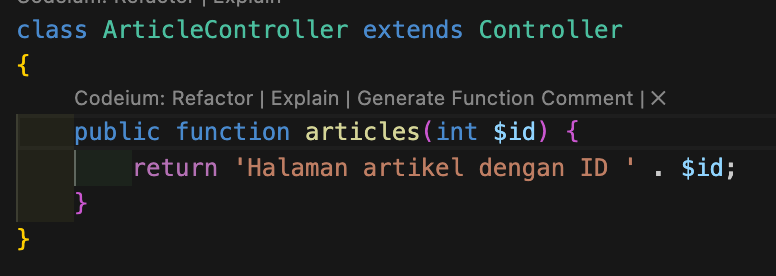
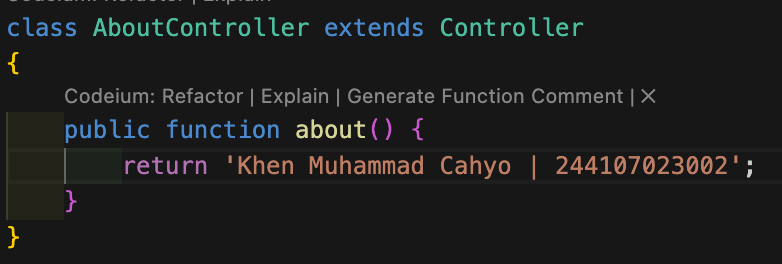
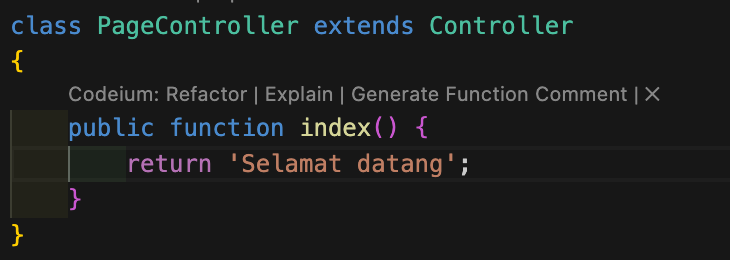
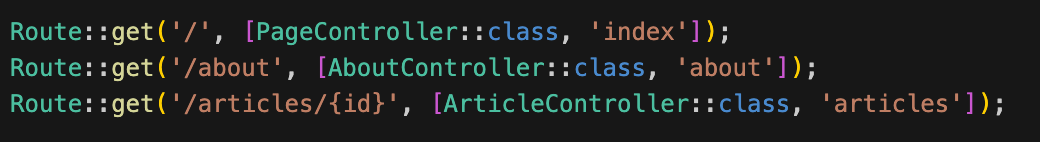
**PRACTICUM CONTROLLER**

* **Creating Controller (a-e)**

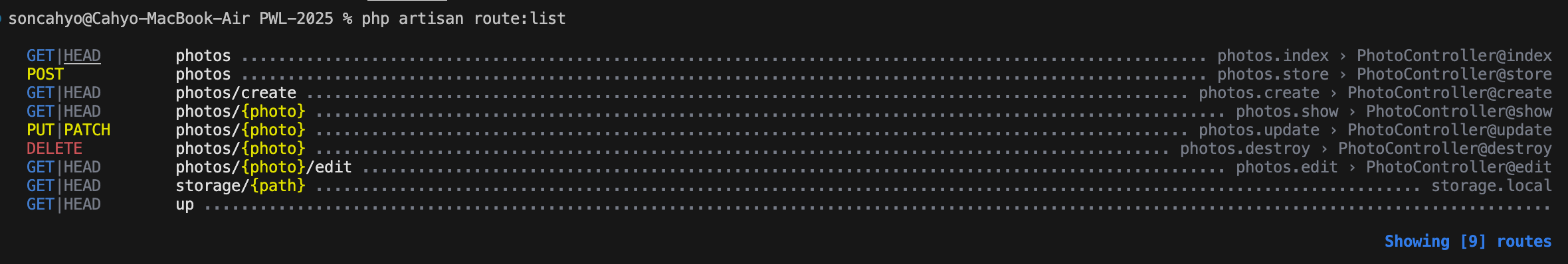


The code defines a route in Laravel that listens for HTTP GET requests at the /hello URL. Instead of using a closure, it maps the route to the hello method in the WelcomeController class using the WelcomeController::class syntax. When the route is accessed, Laravel invokes the hello method within the WelcomeController, which returns the string "Hello World". This demonstrates how to organize application logic in a controller rather than using closures directly in the route definition.

* **Creating Controller (f-g)**



* **Resource Controller (a-d)**

****

****

**PRACTICUM VIEW**

* **Creating View (a-c)**

****

The code defines a route in Laravel that listens for HTTP GET requests at the /greeting URL. When accessed, the route returns a view named hello from the resources/views/hello.blade.php file. The view is passed a variable name with the value 'Andi'. Inside the Blade view, the variable {{ $name }} is used to dynamically display the value of name, so the resulting HTML will show "Hello, Andi" in an <h1> element. This demonstrates how to pass data from a route to a Blade view for dynamic content rendering.

* **View Inside Directory (a-d)**

****

The updated code defines a route in Laravel that listens for HTTP GET requests at the /greeting URL. When accessed, it returns a view named hello located in the blog folder inside the resources/views directory, specifically at resources/views/blog/hello.blade.php. The view is passed a variable name with the value 'Andi', which is used in the Blade template to dynamically display the name. The Blade syntax {{ $name }} in the view will output "Hello, Andi". This demonstrates how to organize views in subdirectories within the resources/views folder and pass data to those views.

* **Showing View From Controller (a-c)**

****

The code defines a Laravel route that listens for HTTP GET requests at the /greeting URL. Instead of using a closure, the route maps to the greeting method in the WelcomeController class. In the greeting method, the view() function is used to return the hello view located in the blog folder (resources/views/blog/hello.blade.php), passing a variable name with the value 'Andi'. Inside the Blade view, the {{ $name }} syntax dynamically displays the name, so the rendered HTML will show "Hello, Andi." This demonstrates how to structure a controller method to handle route logic and pass data to a Blade view.

* **Forwards Data to View (a-c)**

****

The code defines a Blade view in resources/views/blog/hello.blade.php that expects two variables: name and occupation. The greeting method in the controller returns this view and uses the with() method to pass the variables to it—'name' with the value 'Andi' and 'occupation' with the value 'Astronaut'. In the Blade template, the variables are displayed using the {{ $name }} and {{ $occupation }} syntax, which will output "Hello, Andi" and "You are Astronaut" respectively in <h1> tags when the view is rendered. This shows how to pass multiple pieces of data from the controller to a Blade view using the with() method.