Q-1: How to pass Multiple Variable in route?

To pass multiple variables in a Laravel route, you can use the {variable\_name} syntax. For example, the following route

will pass the user\_id and post\_id variables to the controller:

Route::get('users/{user\_id}/posts/{post\_id}', function ($user\_id, $post\_id) {

// Get the user and post from the database

$user = User::find($user\_id);

$post = Post::find($post\_id);

// Return the view

return view('users.posts.show', compact('user', 'post'));

});

Q-2: How to pass variable which can be null in Route?

To pass a variable which can be null in a Laravel route, you can use the {variable\_name?} syntax. This syntax tells

Laravel that the variable is optional, and that it can be null. For example, the following route will pass the brand

variable to the controller, but it is optional, so it can be null:

Route::get('/products/{brand?}', function ($brand) {

// Get the products for the specified brand

if ($brand) {

$products = Product::where('brand', $brand)->get();

} else {

$products = Product::all();

}

// Return the view

return view('products.index', compact('products'));

});

Q-3: Create custom auto using middleware.

Create a new middleware file in the app/Http/Middleware directory. The name of the middleware file should be descriptive

of its purpose. For example, if you are creating a middleware to check if the user is authenticated, you could name the

file AuthenticateMiddleware.php.

The middleware file should extend the Illuminate\Http\Middleware\Middleware class.

In the middleware class, you need to define a method called handle(). This method will be called for every request that

passes through the middleware.

In the handle() method, you can do whatever logic you need to do to determine if the request should be allowed to

continue. For example, you could check if the user is authenticated, or if the request is coming from a trusted IP

address.

If the request is allowed to continue, you can return the next middleware in the stack. Otherwise, you can return a

response that will be sent to the user.

Q-4: Generate Resource Controller for employee.

Step 1: Setup Laravel Project

Step 2: Create Employee Model and Migration

Step 3: Run Migrations

Step 4: Generate Resource Controller

php artisan make:controller EmployeeController --resource

Step 5: Define Routes

Route::resource('employees', EmployeeController::class);

Q-5: In Employee Controller's action Call Middleware.

here are the steps on how to call middleware in an employee controller action in Laravel:

Open the EmployeeController.php file and find the action that you want to call the middleware on.

In the action, add the following code to the top of the method:

use App\Http\Middleware\MyMiddleware;

This will import the middleware class into the controller.

Now you can call the middleware by using the app('mymiddleware') method. For example, the following code will call the

MyMiddleware middleware:

$this->middleware('mymiddleware');

The MyMiddleware middleware will be executed before the controller action is called.

Q-6: How to remove route caching?

To remove route caching in Laravel, you can use the route:clear Artisan command. This command will clear the cache file

that stores the routes.

To use the route:clear command, open a terminal window and navigate to the root directory of your Laravel project. Then,

run the following command:

php artisan route:clear

Q-7: Create Custom Macro For search User

use Illuminate\Database\Eloquent\Macroable;