Laravel - Error Handling

Topics : Laravel
Written on December 25, 2023

In Laravel, error handling is an essential aspect of building robust applications. Laravel provides a comprehensive error handling system that includes exception handling, logging, and customizable error pages.

1. Exception Handling:

Exception handling in Laravel is managed through the `App\Exceptions\Handler` class. This class contains methods for reporting and rendering exceptions.

Reporting Exceptions:

The `report` method is used to log exceptions or send them to external services.

```
public function report(Exception $exception)
{
    if ($exception instanceof CustomException) {
        // Log or report the custom exception
    }

    parent::report($exception);
}
```

Rendering Exceptions:

The `render` method is responsible for converting exceptions into HTTP responses.

```
public function render($request, Exception $exception)
{
    if ($exception instanceof CustomException) {
        return response()->view('errors.custom', [], 500);
    }

    return parent::render($request, $exception);
}
```

2. Logging:

Laravel uses the Monolog library for logging. Log messages can be generated using the `Log` facade.

```
use Illuminate\Support\Facades\Log;

public function someMethod()
{
    try {
        // Code that may throw an exception
    } catch (Exception $e) {
        Log::error('An error occurred: ' . $e->getMessage());
    }
}
```

Log messages are stored in the `storage/logs` directory by default.

3. Custom Error Pages:

Laravel allows you to customize error pages for different HTTP status codes. Error views can be customized in the `resources/views/errors` directory.

For example, to customize the 404 (Not Found) error page, create a `404.blade.php` file in the `errors` directory.

4. Logging Levels:

Laravel supports various logging levels, such as `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `info`, and `debug`. Use appropriate levels based on the severity of the log message.

```
Log::info('This is an informational message');
Log::error('An error occurred');
```

5. Debugging Tools:

During development, Laravel provides convenient debugging tools. The `dump` and `dd` functions can be used to display variable values.

```
dump($variable); // Display variable values
dd($variable);   // Dump and die
```

6. Handling Validation Errors:

In controllers, you can use Laravel's validation features to handle form validation errors.

```
public function store(Request $request)
{
    $request->validate([
        'name' => 'required|string|max:255',
        'email' => 'required|email|unique:users',
        'password' => 'required|min:8',
    ]);
```

```
    // Process the form data
}
```

7. Error Logging Configuration:

Error and log configuration settings can be found in the `config/app.php` and `config/logging.php` files.

8. Environment-based Error Handling:

You can customize error handling based on the application's environment (e.g., development, production) using the `App\Exceptions\Handler` class.

9. Handling 404 Errors:

To customize the behavior for 404 errors, you can use the `render` method in the `App\Exceptions\Handler` class.

```
public function render($request, Exception $exception)
{
    if ($exception instanceof NotFoundHttpException) {
        return response()->view('errors.404', [], 404);
    }

    return parent::render($request, $exception);
}
```