Laravel - Errors and Logging

Topics : Laravel
Written on December 25, 2023

In Laravel, handling errors and logging is crucial for monitoring and debugging applications. Laravel provides powerful tools for error handling and logging to help developers identify and resolve issues. Here are key aspects related to errors and logging in Laravel:

1. Configuration:

Error handling and logging configurations can be found in the config/app.php and config/logging.php files. In the .env file, you can set the APP_ENV variable to control the application's environment (e.g., development, production).

2. Error Pages:

Laravel includes customizable error views for 404 (Not Found) and 500 (Internal Server Error) pages. You can customize these views in the resources/views/errors directory.

3. Exception Handling:

The App\Exceptions\Handler class is responsible for handling exceptions that occur during the request lifecycle. The report method is used for logging exceptions, and the render method is used to customize the error responses.

4. Logging:

Laravel uses the Monolog library for logging. By default, logs are stored in the storage/logs directory. You can customize logging channels, levels, and handlers in the config/logging.php file.

5. Logging Levels:

Laravel supports various logging levels such as emergency, alert, critical, error, warning, notice, info, and debug. You can use these levels based on the severity of the log message.

```
// Example log message
Log::error('An error occurred');
```

6. Logging to Custom Channels:

You can define custom logging channels in the config/logging.php file. Channels can log messages to different locations, such as files, the system log, or external services.

```
// Example custom channel
'channels' => [
```

```
    'custom-channel' => [
        'driver' => 'daily',
        'path' => storage_path('logs/custom.log'),
        'level' => 'debug',
    ],
],
```

7. Log Storage and Rotation:

Laravel supports log rotation to prevent log files from becoming too large. You can configure log rotation in the config/logging.php file.

8. Logging in Controllers and Services:

You can use the Log facade to log messages from within your controllers, services, or other parts of your application.

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

public function someMethod()
{
    Log::info('This is an informational message');
}
```

9. Debugging with dd and dump:

During development, you can use the dd (dump and die) and dump functions for quick debugging. These functions provide insights into variables and data structures.

```
dd($variable); // Dump and die
dump($variable); // Dump without terminating the script
```